digital

VAX–11 PATCH

Utility Reference Manual

Order No. AA-H785A-TE

VAX11

March 1980

This document describes how to use the VAX-11 Image File Patch Utility (PATCH). It provides necessary information for examining and modifying shareable images, device-driver images, and executable images.

VAX-11 PATCH
Utility Reference Manual

Order No. AA-H785A-TE

SUPERSESSION/UPDATE INFORMATION: This is a new document for this release.

OPERATING SYSTEM AND VERSION: VAX/VMS V02

SOFTWARE VERSION: VAX/VMS V02

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation · maynard, massachusetts

First Printing, March 1980

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

CONTENTS

CONTENTS

CONTENTS

CONTENTS

PREFACE

MANUAL OBJECTIVES

This manual describes the VAX-11 Image File Patch Utility (PATCH). It contains general usage information about PATCH, the syntax rules for using PATCH correctly, and reference information about PATCH commands.

INTENDED AUDIENCE

This manual is intended to be used by experienced system programmers who are writing or modifying device driver images, shareable images, or executable images. Familiarity with a patching utility is not required; this manual provides introductory information on PATCH and on PATCH-specific concepts and terminology.

STRUCTURE OF THIS DOCUMENT

This manual contains eight chapters and two appendixes:

- Chapter 1 contains a functional overview of the PATCH utility. It introduces basic concepts in PATCH and illustrates a simple PATCH session. The remainder of the chapter summarizes the PATCH commands by functional category.
- Chapter 2 discusses PATCH input and output files.
- Chapter 3 explains how to run PATCH and how PATCH command qualifiers affect the output files.
- Chapter 4 provides an overview of symbol usage in PATCH -- the various symbols recognized by PATCH, how to use symbols to reference locations, and how PATCH performs symbol and value translations.
- Chapter 5 describes the two types of patch area available in PATCH, including when to choose one type of patch area over the other, how to access patch area, and the commands that affect patch area.
- Chapter 6 provides a complete introduction, with examples, to PATCH's entry and display modes. In addition, this chapter contains a functional summary of the modes.
- Chapter 7 describes the syntax rules PATCH follows when processing commands.
- Chapter 8 contains detailed descriptions of all PATCH commands; the commands are listed in alphabetical order.

- Appendix A contains an alphabetical listing of the PATCH commands in their required format. Also included are the qualifiers that can be specified with the commands and a brief functional definition of each command.

- Appendix B lists the error messages produced by PATCH. Each error message is accompanied by a brief definition of the error and, in some cases, a suggested user response is supplied.

ASSOCIATED DOCUMENTS

Other manuals that may be of use are:

- VAX-11 Symbolic Debugger Reference Manual
- VAX-11 Linker Reference Manual

CONVENTIONS USED IN THIS DOCUMENT

The following conventions are used in this manual:

PATCH> EXAMINE/INSTRUCTION 606 00000606: 'TSTL R4'	In all examples, the output lines and prompting characters printed or displayed by the system are in black ink; all user-entered commands are in red ink.
CTRL/x	The expression "CTRL/x" means that you enter a control character by pressing the appropriate key while you hold down the CTRL key.
new-instruction . . .	A vertical ellipsis indicates that not all the data the system would normally display or a user would normally enter is shown.
eco-level,...	A horizontal ellipsis indicates that additional parameters, values, or information can be entered.
[symbol-name]	Square brackets indicate that the enclosed expression is optional.

CHAPTER 1

OVERVIEW

After you have compiled or assembled and linked a program, you may find it necessary to make one or more changes to the code. There are two ways to change the code. One way is to edit the source program, recompile or re-assemble, relink, and run the new version. Sometimes, however, the source program is not readily available for editing or the time required to make the changes and create an executable image is far too lengthy. A second way, then, is to use the VAX-11 Image File Patch Utility (PATCH).

PATCH provides an extensive set of commands that lets you make changes directly to the image file and then run the new version without recompiling or re-assembling and relinking. Furthermore, PATCH creates a journal file in which all PATCH commands used are recorded. This file provides an easy way to keep track of the changes and attempted changes made to an image file.

You can use PATCH to edit an image file during an interactive terminal session. You can also enter PATCH commands in a command procedure then submit the command procedure for interactive or batch mode processing.

PATCH also provides several features to help you find and correct erroneous code. These features include:

- Symbolic referencing of locations
- Patch area to store additional data and instructions
- Entry and display modes to control the environment in which PATCH accepts your commands and displays its output

This chapter provides a brief overview of the PATCH utility and directs you to the appropriate chapters that contain more detailed descriptions of the particulars of PATCH. The colored pages at the end of this chapter list the PATCH commands by function.

1.1 WHEN TO USE PATCH

PATCH is intended to be used to maintain executable image files, shareable image files, and device-driver image files written by users in any language supported by the VAX/VMS operating system. PATCH is not intended for customer use on DIGITAL software. Any revisions applied to this software, except those contained in DIGITAL-supplied auto update kits, may make it difficult for you to install future software updates. Furthermore, application of patches to VAX/VMS software which are not supplied by DIGITAL, invalidates the warranty on the DIGITAL-supplied VAX/VMS software.

1.2 PATCH AND THE VAX-11 SYMBOLIC DEBUGGER

In many cases, PATCH can be used with the VAX-11 Symbolic Debugger. The debugger lets you step through your image file and make temporary changes to the code. Once you have decided on the new code, PATCH makes the changes permanent in the new version of the image file. Keep in mind, however, that the source code will not contain your changes until you enter them. For a detailed description of the function and capabilities of the VAX-11 Symbolic Debugger, see the VAX-11 Symbolic Debugger Reference Manual.

1.3 INPUT AND OUTPUT FILES

You can use PATCH to make changes to an image file during an interactive terminal session or by submitting a command procedure for interactive or batch mode processing.

During interactive patching, input consists of an input image file, which is the file you want updated, and PATCH commands that, when executed, perform the requested changes.

During patching by means of a command procedure, the only input is a PATCH command procedure, which can be created during a PATCH session or by using a text editor. The command procedure contains the name of the image file to be patched and the PATCH commands you want applied to the image file.

When you patch an image file, PATCH can produce the following three output files:

File	Description
Output image file	An updated image file. This file must be explicitly created using the UPDATE command.
Journal file	An ASCII file containing a record of the PATCH session. This file is automatically created by PATCH.
Command procedure	A file containing all successful PATCH commands that can subsequently be used as input to PATCH. This file must be explicitly created using the CREATE command.

Refer to Figure 1-1 to see how PATCH updates an image file and read Chapter 2 for a complete description of PATCH's input and output files.

1.4 USING PATCH

This section contains general information necessary for using PATCH efficiently and effectively. This information includes how to run PATCH and how to apply a patch to an image file; it also includes information on symbol usage, patch area, and the entry and display modes.

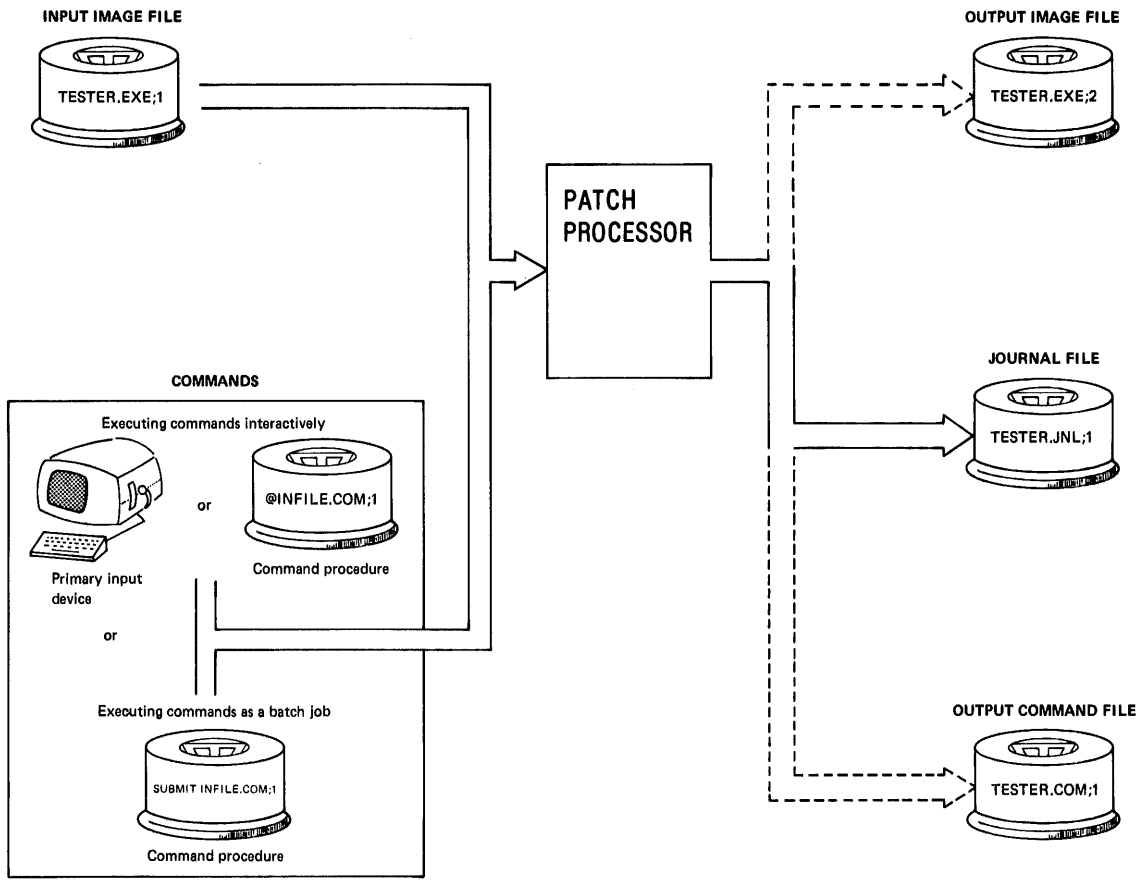

Figure 1-1 PATCH Input and Output

1.4.1 Starting and Stopping PATCH

The following DCL command line invokes PATCH for an interactive terminal session:

```
$ PATCH file-spec <RET>
```

The file-spec indicates the name of the input image file that is to be patched.

After you press the RETURN key, PATCH displays an introductory message, then indicates that it is ready to accept commands by issuing the following prompt:

```
PATCH>
```

Once you have initiated PATCH, you can interrupt program execution by typing CTRL/Y (echoed as ^Y). Typing this command returns control to the VAX/VMS command interpreter. If you decide that you still want to execute PATCH, type the CONTINUE command.

To end a PATCH session, type either the EXIT command or CTRL/Z.

Chapter 3 provides a complete description, with examples, on starting and stopping PATCH, including how to submit command procedures for interactive and batch mode processing.

1.4.2 Applying Patches

Whenever you apply a patch to an image file, whether the patch is to be applied interactively or through a command procedure, the following types of PATCH commands should be used:

- A patch initiator -- the SET ECO command that defines a unique Engineering Change Order (ECO) level between 1 and 128, inclusive for each patch. The ECO level serves as an identifier for the patch. (See Chapter 8.)

- PATCH commands that examine and modify the image file. (See Chapter 8.)

- A patch terminator -- the UPDATE command that sets the ECO level and applies the patch to the image file. (See Chapter 8.)

These commands must be used in the order shown. They can be repeated as many times as necessary during one execution of PATCH.

The following example typifies a patch for the image file named NEGATION.EXE;1:

```
PATCH> SET ECO 1
PATCH> EXAMINE/INSTRUCTION 606
00000606:   TSTL R4
PATCH> REPLACE/INSTRUCTION 606 = 'TSTL R4'
NEW> 'TSTL R5'
NEW> EXIT
old:   00000606:   TSTL R4
new:   00000606:   TSTL R5
PATCH> UPDATE
%PATCH-I-WRTFIL, updating image file DB1:[GARTH]NEGATION.EXE;2
```

In the above example, the ECO level is defined as 1. The EXAMINE command requests that location 606 be displayed as a VAX-11 MACRO instruction. The REPLACE command then changes the instruction in location 606 to a different instruction. The UPDATE command sets ECO level 1 and applies the patch to NEGATION.EXE.

For information on individual PATCH commands, see Chapter 8.

1.4.3 Using Symbols

To change the code in an image file, you need to identify the locations that contain the erroneous instructions or data. That is, you must identify the virtual addresses within the image file. You can locate virtual addresses using one of two ways:

1. Obtain a listing and a map of the image file; the listing gives you the relative offsets for each module in your image file, and the map tells you where each module starts in virtual memory.

2. Locate the addresses by means of symbolic names defined in the image file when it was created.

Using the second method, upon invoking PATCH, you can reference locations by specifying the symbolic names assigned to them and PATCH will resolve the symbolic references using the symbol table and scope.

1.4.3.1 **PATCH's Symbol Table** - PATCH provides a symbol table in which all symbols are placed. (See Chapter 4 for information on passing symbols to PATCH.) Any symbol contained in this table can be used to access one or more locations. PATCH will display an error message when you attempt to use a symbol not contained in this table.

To determine the status of the symbols in the symbol table, type the SHOW MODULE command. Then if you want to add or delete specific symbol information, type the SET MODULE or CANCEL MODULE command. These commands are fully described in Chapter 8.

1.4.3.2 **Definition of Scope** - When a symbol is multiply defined (that is, when it represents two or more memory locations) and the symbol is available to PATCH, you must indicate which memory location you mean to reference when you use that symbol in a command. You do so by specifying the "scope" of the symbol. The scope is the name of the module and optionally routine in which the symbol appears. To establish the correct scope, type the SET SCOPE command or prefix the symbol with the module or routine name in which it is contained. For more information on the SET SCOPE command, see Chapter 8.

1.4.3.3 **Symbols and Pathnames** - PATCH allows you to use symbols and pathnames to find and to define specific memory locations within your image file. The symbols PATCH recognizes are:

- Global symbols
- Local symbols
- Module names, program section names, and routine names
- Universal symbols
- Symbolic instruction labels
- Symbols defined with the DEFINE command

Some symbols, such as two local symbols or a local and global symbol, may represent more than one memory location in an image file. For example, the local symbol XYZ could represent a memory location in module A and another memory location in module B. To resolve the ambiguity of symbol XYZ, PATCH allows the use of pathnames to reference locations.

A pathname is a complete and unambiguous symbolic translation of a location. That is, a pathname defines the scope, symbol name, and offset value, in the following format:

scope\symbol-name±offset-value

If the symbol is unique, the pathname consists of the symbol name and, possibly, an offset value; the scope becomes unnecessary.

For a detailed description of the various symbols that PATCH recognizes, see Chapter 4.

1.4.3.4 **Passing Symbols** - To ensure that symbols are available for your use, you must include certain information at assembly or compile and link time. For example, to pass local symbols in a MACRO program, include the /ENABLE=DEBUG qualifier when you assemble the program and the /DEBUG qualifier when you link the program.

Read the appropriate language user's guide to determine which qualifiers are required to pass symbol information to PATCH and read the VAX-11 Linker Reference Manual to determine how to pass universal symbols to PATCH.

1.4.4 Using Patch Area

In many cases, applying a patch to an image file requires adding lines of code to the file. A convenient technique for adding code is to insert such code into storage space called patch area.

There are two types of patch areas: default patch area and user-defined patch area.

PATCH automatically inserts additional code into the current patch area (either default or user defined) when there is not enough space at the requested location to insert the code. PATCH then inserts branch instructions to and from the patch area to maintain the correct flow of program execution.

You can also request that data be placed in the current patch area by issuing the DEPOSIT/PATCH_AREA command. Keep in mind, however, that PATCH does not generate branch instructions to and from the patch area when you use the DEPOSIT/PATCH_AREA command; you must keep track of and document the correct flow of program execution.

For complete information on when and how to use default patch area and user-defined patch area, see Chapter 5.

1.4.5 Using Entry and Display Modes

To make referencing locations in an image file more convenient, PATCH provides four major categories of addressing modes, also called entry and display modes. These four mode categories are: context, radix, length, and symbol search. Table 1-1 provides a brief functional definition of each mode category.

When you invoke PATCH, the modes are set to: NOASCII, NOINSTRUCTION, SYMBOLS, HEXADECIMAL, LONG, NOGLOBALS, and SCOPE. To change these modes, issue the SET MODE and CANCEL MODE commands or issue an alternative mode qualifier on the command line.

Entry and display modes are discussed in Chapter 6, and in Chapter 8 under the SET MODE command.

Table 1-1
Functional Division of the Entry and Display Modes

Category	Mode	Function
Context	[NO]ASCII [NO]INSTRUCTION [NO]SYMBOLS	Controls whether PATCH accepts and displays data as ASCII text or VAX-11 MACRO instructions, and displays addresses in symbolic representation
Radix	OCTAL DECIMAL HEXADECIMAL	Controls the base in which PATCH accepts and displays data and addresses
Length	BYTE WORD LONG	Controls the size in which PATCH accepts and displays data
Symbol Search	[NO]GLOBALS [NO]SCOPE	Controls how PATCH evaluates symbol and value translations

1.5 SAMPLE PATCH SESSION

This section uses an example to illustrate an interactive PATCH session. The following VAX-11 MACRO program prompts for a decimal integer, then displays the negation of the integer.

```
    0000     1          .TITLE NEGATION - Calculates negation of decimal integer
    0000     2          .IDENT /01/
    0000     3
    0000     4          .SUBTITLE - Pure Data
00000000     5          .PSECT RO_DATA,NOWRT,NOEXE,LONG
    0000     6
    0000     7 PROMPT:  .ASCII  /Enter decimal integer/
    000C
    0016     8
    0016     9 PROMPT_LEN = .-PROMPT
    0016    10 RESULT_FAO:
    0016    11          .ASCID \The negation is !SL.\
    0024
    0030
    0032    12
    0032    13          .SUBTITLE - Impure Data
00000000    14          .PSECT RW_DATA,NOEXE,LONG
    0000    15
    0000    16 BUFFER_SIZE     =       132              !Buffer size
    0000    17 BUFFER_DESC:
    0000    18          .LONG   BUFFER_SIZE,BUFFER       !Descriptor for buffer
    0008    19
    0008    20 BUFFER:
    0008    21          .BLKB   BUFFER_SIZE              !Buffer to get line
    008C    22
    008C    23 INTEGER:
    008C    24          .BLKL   1                        !Input integer
    0090    25 RESULT:
    0090    26          .BLKL   1                        !Result of operation
    0094    27
    0094    28          .SUBTITLE       Define RAB and FAB for terminal I/O
    0094    29
    0094    30 TRMFAB:  $FAB    FNM=<SYS$INPUT>,RAT=CR   !FAB for terminal
    00E4    31 TRMRAB:  $RAB    FAB=TRMFAB,UBF=BUFFER,USZ=BUFFER_SIZE,ROP=PMT-
    00E4    32                  PBF=PROMPT,PSZ=PROMPT_LEN       !RAB for terminal
    0128    33
    0128    34          .SUBTITLE       PROGRAM ENTRY POINT
00000000    35          .PSECT RO_CODE,EXE,NOWRT
    0000    36          .ENTRY ENTRY_POINT,0
```

```
0002    37
0002    38          $OPEN   FAB=TRMFAB                ;Open terminal file
000F    39          $CONNECT        RAB=TRMRAB        ;Connect record stream
001C    40          $GET    RAB=TRMRAB                ;Get input from terminal
0029    41
0029    42          PUSHAL  INTEGER                   ;Push integer address on stack
002F    43          PUSHL   RAB$L_RBF+TRMRAB          ;Push starting address of
0035    44                                            ;input on stack
0035    45          MOVZWL  RAB$W_RSZ+TRMRAB,-(SP)    ;Push length of input on stack
003C    46          CALLS   #3,G^LIB$CVT_DTB          ;Convert to binary
0043    47          MOVL    INTEGER,RESULT            ;Calculate negation
004E    48
004E    49 ;Determine length and contents of result
004E    50
004E    51          $FAO_S  CTRSTR=RESULT_FAO,-
004E    52                  OUTLEN=TRMRAB+RAB$W_RSZ,-
004E    53                  OUTBUF=BUFFER_DESC,-
004E    54                  P1=RESULT

006D    55
006D    56 ;Output result
006D    57
006D    58          MOVAB   BUFFER,TRMRAB+RAB$L_RBF
0078    59          $PUT    RAB=TRMRAB
0085    60          MOVZBL  #SS$_NORMAL,R0
0089    61          RET
008A    62          .END    ENTRY_POINT
```

The program will assemble and link, but when executed will produce erroneous results, as follows:

```
$ MACRO/ENABLE=DEBUG NEGATION.MAC;1
$ LINK/DEBUG NEGATION.OBJ;1
$ RUN/NODEBUG NEGATION.EXE;1
Enter decimal integer 25
The negation is 25.
$
```

After locating the incorrect code, by use of the VAX-11 Symbolic Debugger or simply by examining the listing, you can invoke PATCH to make a permanent change in the image file as shown below. The numbers to the right of the example are keyed to the explanations that follow the example.

```
$ PATCH NEGATION.EXE;1

  PATCH VERSION 2.0 15 MAR 80                                          (1)

PATCH> SET ECO 1                                                       (2)
PATCH> SET MODULE/ALL                                                  (3)
PATCH> SET SCOPE NEGATION                                              (4)
PATCH> SET MODE INSTRUCTION                                            (5)
PATCH> EXAMINE RO_CODE+43
NEGATION\RO_CODE+43         MOVL L^NEGATION\INTEGER,L^NEGATION\RESULT  (6)
PATCH> REPLACE NEGATION\RO_CODE+43
OLD> 'MOVL L^NEGATION\INTEGER L^NEGATION\RESULT'
OLD> EXIT
NEW> 'MNEGL L^NEGATION\INTEGER L^NEGATION\RESULT'
NEW> EXIT
old: NEGATION\RO_CODE+43:  MOVL  L^NEGATION\INTEGER,L^NEGATION\RESULT
new: NEGATION\RO_CODE+43:  MNEGL L^NEGATION\INTEGER,L^NEGATION\RESULT
PATCH> UPDATE
%PATCH-I-WRTFIL, updating image file  DB1:[GARTH]NEGATION.EXE;2        (7)
PATCH> EXIT                                                            (8)
$
```

❶ Invoke PATCH -- Invoke PATCH for an interactive terminal session by typing the DCL command line PATCH NEGATION.EXE;1. PATCH, in return, displays an introductory message on your terminal. This message indicates the version of PATCH that you are using and its release date.

❷ Define an ECO level -- The SET ECO command defines the ECO level for the ensuing PATCH. Note that the ECO level is not set until you issue the UPDATE command.

❸ Alter the symbol status -- The SET MODULE/ALL command inserts all local symbol information into the symbol table provided that you followed the rules for passing local symbols at compile or assembly and link time.

❹ Alter the scope status -- The SET SCOPE command changes the contents of the scope. In this case, the scope has been set to the module named NEGATION.

❺ Alter entry and display mode status -- The SET MODE command sets the INSTRUCTION mode, causing PATCH to display the contents of the specified locations as VAX-11 MACRO instructions.

❻ Change the code -- A number of PATCH commands can be used to modify code. Each command alters the code in a unique fashion. In this example, the REPLACE command was used to overwrite the existing instruction with a new, correct instruction.

❼ Set the ECO level and update the image file -- After you correct the code, issue the UPDATE command to set the specified ECO level and to create a new image file consisting of the original image file and the patch.

❽ Exit from PATCH -- Type the EXIT command to end the PATCH session and return control to the VAX/VMS command interpreter.

Now run the program NEGATION.EXE;2 and it will execute properly.

```
$ RUN NEGATION.EXE;2
Enter decimal integer 37
The negation is -37.
$
```

1.6 COMMANDS IN PATCH

This section describes the general format of the PATCH commands. It also provides a functionally arranged summary of these commands.

1.6.1 Command Line Format

You issue PATCH commands to examine, verify, and modify locations in an image file. A PATCH command takes the following general format:

```
command [/qualifier(s)]    [parameter(s)]    [!comment]
```

command

Specifies the name of a PATCH command.

/qualifier(s)

Specifies the name of one or more qualifiers that modify the intent of the command. A qualifier can be one of the following:

- Alignment qualifier
- Mode qualifier
- Other qualifier

Mode qualifiers are briefly described under the "Mode Qualifiers" heading of each command in Chapter 8. For a complete description of the mode qualifiers, see Chapter 6.

You can enter multiple qualifiers after a single command provided that you precede each qualifier with the slash character (/).

parameter(s)

Specifies the target of the command, such as a range of memory locations.

!comment

Specifies a comment. PATCH ignores all data starting with an exclamation character (!).

See Chapter 7 for a complete description of the rules for entering commands.

1.6.2 Functional Summary of PATCH Commands

This section summarizes the uses of the PATCH commands. The commands are functionally grouped in the following five tables:

- Commands for Creating Patches (Table 1-2)
- Commands for Manipulating Patch Area (Table 1-3)
- Commands for Examining and Modifying Locations (Table 1-4)
- Commands for Controlling PATCH Modes (Table 1-5)
- Commands for Expressing Symbols and Pathnames (Table 1-6)

These tables briefly describe each command. See Chapter 8 for detailed command descriptions.

Table 1-2
Commands for Creating Patches

Command	Description
CHECK ECO	Verifies the application of the patches associated with the specified ECO levels
CHECK NOT ECO	Verifies that the specified ECO levels are available for use
CREATE	Opens a command procedure to which all subsequent, successful PATCH commands are written
SET ECO	Defines an ECO level for the ensuing patch
UPDATE	Sets the ECO level and applies the patch to the image file

Table 1-3
Commands for Manipulating Patch Area

Command	Description
ALIGN	Aligns the first free byte of the current patch area on the specified boundary and defines a symbol for that address
CANCEL PATCH_AREA	Cancels the use of the current user-defined patch area and reverts to the use of the default patch area
DEPOSIT/PATCH_AREA	Deposits data or VAX-11 MACRO instructions in the current patch area
INSERT/INSTRUCTION	Inserts one or more VAX-11 MACRO instructions into the current patch area
SET PATCH_AREA	Establishes a user-defined patch area as the current patch area
SHOW PATCH_AREA	Displays the string descriptor of the current patch area

Table 1-4
Commands for Examining and Modifying Locations

Command	Description
DELETE	Verifies the current contents of one or more consecutive locations, then replaces the contents with NOP instructions or zeros
DEPOSIT	Deposits new values into one or more consecutive locations
EXAMINE	Displays the contents of one or more locations
INSERT/INSTRUCTION	Verifies the current contents of a location, then inserts one or more VAX-11 MACRO instructions after that location
REPLACE	Verifies the contents of one or more consecutive locations, then replaces the contents with new contents
VERIFY	Verifies the contents of one or more consecutive locations

Table 1-5
Commands for Controlling PATCH Modes

Command	Description
CANCEL MODE	Cancels the current mode settings and reinstates the initial mode settings
SET MODE	Sets new entry and display modes
SHOW MODE	Displays the current mode settings

Table 1-6
Commands for Expressing Symbols and Pathnames

Command	Description
CANCEL MODULE	Removes local symbols and program section names defined in the specified module from PATCH's symbol table
CANCEL SCOPE	Cancels the current symbolic scope and reinstates the <null> scope
DEFINE	Equates a symbolic name with a value
EVALUATE	Displays the current symbolic definition or definitions of a value
SET MODULE	Enters local symbols and program section names defined in the specified module into PATCH's symbol table
SET SCOPE	Defines the current symbolic scope
SHOW MODULE	Displays all the modules in the image file being patched and indicates whether the local symbols contained in those modules are entered in PATCH's symbol table
SHOW SCOPE	Displays the current symbolic scope

CHAPTER 2

PATCH INPUT AND OUTPUT FILES

Input to PATCH is (1) an input image file that you want to update and (2) PATCH commands that perform the requested update operations. You can issue PATCH commands directly from your terminal or you can enter the commands in a command procedure in the order of their execution then submit the command procedure for interactive or batch mode processing.

PATCH creates one or more of the following files as output:

- Output image file
- Journal file
- Command procedure

The output image file and command procedure are optional files. You must request them for PATCH to create them. PATCH automatically creates a journal file.

This chapter describes PATCH input and output files. See Chapter 8 for PATCH command descriptions.

2.1 INPUT IMAGE FILE

The input image file is the file you want to update. Because PATCH is not a language-dependent utility, you can use it to update an image file written in any language supported by the VAX/VMS operating system. The input image file can be a shareable image, device driver image, or executable image. The only restriction is that the input image file must be created by the VAX-11 Linker.

PATCH never alters the input image file. When you invoke PATCH, it creates a copy of the input image file and then incorporates your changes into the copy. You can thus test the new image file and, if it does not run correctly, still have the original file to re-evaluate.

2.1.1 Shareable Images

A shareable image is an image that does not have a starting address and must be linked with one or more object modules to produce an executable image.

You must take the following into consideration when patching shareable images:

- You can specify only universal symbols when patching a shareable image (see Section 4.1.4)
- You should use a user-defined patch area rather than the default patch area to store additional instructions and data (see Section 5.2.1)

For additional information on shareable images, see the VAX-11 Linker Reference Manual.

2.1.2 Device Driver Images

A device driver image is a set of routines and tables used by the operating system to process an I/O request for a particular device type. To patch a device driver image, you should use a user-defined patch area rather than the default patch area to store additional instructions and data. (See Section 5.2.1.)

2.1.3 Executable Images

An executable image is the most common type of image. An executable image is one that can be run by a process. When run, an executable image is read from a file for execution by a process. Patching an executable image requires no special rules or considerations. Simply use PATCH as described in this manual. For additional information on executable images, see the VAX-11 Linker Reference Manual.

2.2 OUTPUT IMAGE FILE

The output image file is an updated input image file. It consists of 1) all the changes performed by the commands, and 2) the original data that was not altered. An output image file is always created after you enter the UPDATE command. If you fail to enter the UPDATE command, PATCH does not create an output image file.

During a single execution of PATCH, you can apply several patches to an image file. As you terminate each patch (by issuing the UPDATE command), PATCH updates the output image file. The first UPDATE command creates the output image file; subsequent UPDATE commands (issued during the current PATCH session) overwrite this file.

By default, the output image file specification consists of the same file name and type as the input image file name and type, and a version number 1 greater than the highest version of the input image file. Also by default, the file is created in your process's current device and directory.

To change the default file specification for the output image file, use the /OUTPUT command qualifier with the PATCH command. This qualifier is described in Section 3.2.

2.3 JOURNAL FILE

The journal file provides a complete record of all PATCH commands issued during the editing of an input image file. A journal file entry is created every time a PATCH command is processed. The journal file, then, provides a record not only of changes actually made, but also unsuccessful attempts to edit the input image file.

A journal file is maintained for every PATCH session. If a journal file already exists for a particular file, the new journal entries are appended to it. Otherwise, PATCH creates a journal file with the same file name as the input image file name, a file type of JNL and a version number of 1. Also by default, the file is created in your process's current default device and directory.

To change the default file specification for the journal file, use the /JOURNAL command qualifier with the PATCH command. This qualifier is described in Section 3.2.

Entries in a journal file are written as ASCII text, thereby allowing you to examine the contents of the journal file simply by issuing the DCL command PRINT or TYPE.

Figure 2-1 illustrates a typical journal file that contains two PATCH sessions.

2.4 COMMAND PROCEDURE

You can use a command procedure to apply patches to an image file. A command procedure is a sequence of commands and, optionally, input data that performs a specific action, similar to a program (see the VAX/VMS Guide to Using Command Procedures).

Used with PATCH, a command procedure is a file of PATCH commands that, when the command procedure is executed, will apply patches to a particular image file. Usually, you create a command procedure to facilitate the task of patching multiple copies of a particular image file. Instead of manually patching each file, you can submit a command procedure for interactive or batch mode processing.

See Chapter 3 for complete information on submitting command procedures for interactive and batch mode processing.

There are two ways to create command procedures. One way is to use the PATCH command CREATE. The second way is to use an interactive text editor. Both ways are described below.

2.4.1 Using the CREATE Command to Create Command Procedures

After you invoke PATCH, issue the CREATE command to request that a command procedure be created and opened. PATCH automatically inserts the name of the input image file as the first entry in the command procedure. All subsequent, successful PATCH commands are then written into this file. See Chapter 8 for a complete description of the CREATE command.

```
   PATCH VERSION 2.0 15 MAR 80

IMAGE FILE BEING PATCHED:          "DB1:[EAGLE]AVERAGE.EXE;2"
JOURNAL FILE:                      "DB1:[EAGLE]AVERAGE.JNL;1"
DATE/TIME OF PATCH                 20-MAR-1980 12:35:05.53

PATCH>CREATE AVERCMD
COMMAND FILE:                      "DB1:[EAGLE]AVERCMD.COM;1"
PATCH)SET ECO 1
PATCH>SET MODE NOSYMBOLS,INSTRUCTION
PATCH>SHOW MODE
modes: nosymbols, instruction, noascii, scope, noslobals, hexadecimal, long
PATCH>EXAMINE 627
00000627:     PUSHAL   L^00000224
PATCH>INSERT/INSTRUCTION
LOC>627
OLD>'PUSHAL L^00000224'
NEW>'DECL (R0)'
NEW>EXIT
old:      00000627:    PUSHAL L^00000224
new:      00000627:    BRW    00007800
new:      0000062A:    NOP
new:      0000062B:    NOP
new:      0000062C:    NOP
new:      00007800:    PUSHAL L^00000224
new:      00007806:    DECL   (R0)
new:      00007808:    BRW    0000062D
PATCH>UPDATE
%PATCH-I-WRTFIL, updating image file DB1:[EAGLE]AVERAGE.EXE;3
PATCH>EXIT

  PATCH VERSION 2.0 15 MAR 80

IMAGE FILE BEING PATCHED:          "DB1:[EAGLE]AVERAGE.EXE;3"
JOURNAL FILE:                      "DB1:[EAGLE]AVERAGE.JNL;1"
DATE/TIME OF PATCH:                21-MAR-1980 15:38:11.36

PATCH)SET ECO 2
PATCH>SET MODE INSTRUCTION
PATCH>EXAMINE AVERAGE+69
AVERAGE+69:      TSTL R4
PATCH>DELETE/NOSYMBOLS 669 = 'TSTL R4'
old:      00000669:    TSTL R4
new:      00000669:    NOP
new:      0000066A:    NOP
new:      0000066B:    NOP
PATCH>UPDATE
%PATCH-I-WRTFIL, updating image file DB1:[EAGLE]AVERAGE.EXE;4
PATCH>EXIT
```

Figure 2-1 Sample Journal File

As PATCH writes commands into the command procedure, all symbolic references are changed to absolute values. This happens because the steps that transform a program into an image file do not necessarily include the passing of symbol information to the image file. For example, if an image file has been linked without the /DEBUG qualifier, neither local nor global symbol information appears in the image's symbol table. Hence, when using PATCH's CREATE command to create a command procedure, PATCH changes all symbolic references to absolute values in case the image file to which the command procedure is to be applied lacks symbol information.

PATCH also automatically abbreviates all commands in command procedures created using CREATE, to conserve space.

2.4.2 Using Text Editors to Create Command Procedures

You can use a text editor, such as SOS, to construct command procedures. When you do so, the first entry in the command procedure must be the name of the input image file that is to incorporate the patches. Then enter PATCH command lines to the file in the order that you want PATCH to process them. For example:

```
$ EDIT TEST.COM
Input:  DB1:[TERHUNE]TEST.COM;1
00100  AVERAGE.EXE;3
00200  SET ECO 11
00300  DEPOSIT/ASCII 506 = 'RUN'
00400  UPDATE
00500  EXIT
00600  ^Z
*E
```

This file, named TEST.COM, contains a patch, identified by ECO level 11, to be applied to the image file AVERAGE.EXE;3. To process the command procedure interactively, type:

```
$ PATCH @TEST
```

When using a text editor to create a command procedure, you can include symbolic references in the command procedure, if either of the following conditions is met:

1. The symbolic names have been previously defined in the image file to which the command procedure is to be applied

2. If you use the DEFINE command, that you are processing the entire command procedure and not selected patches contained in the command procedure

You can also truncate commands to their accepted abbreviations when creating command procedures with text editors; however, for better comprehensibility and legibility, you should write the commands in long form.

CHAPTER 3

INVOKING AND TERMINATING PATCH

You can use PATCH in three ways: (1) by invoking an interactive terminal session, (2) by submitting a command procedure for interactive execution, or (3) by submitting a command procedure for batch mode processing. This chapter describes how to invoke and terminate PATCH for each of these execution modes. It also discusses the command qualifiers that PATCH recognizes. These qualifiers determine the file specifications for the journal file and output image file, and let you selectively apply individual patches to an image file.

3.1 HOW TO INVOKE PATCH

You can invoke PATCH by responding to the DCL prompt in any one of three ways:

1. $ PATCH [/command-qualifier(s)] file-spec

2. $ PATCH [/command-qualifier(s)] @file-spec

3. $ SUBMIT file-spec

The first method lets you invoke PATCH for an interactive terminal session; the second method lets you execute a command procedure interactively; and the third method lets you submit a command procedure for batch mode processing.

For each of the above methods of invoking PATCH, if you fail to include the file specification on the initial command line, PATCH (or DCL) will prompt for it.

You can also specify command qualifiers when you invoke PATCH. These qualifiers control the characteristics of the output files. See Section 3.2 for complete information on the PATCH command qualifiers.

3.1.1 Invoking PATCH for Interactive Execution

When you invoke PATCH for interactive execution, the file specification indicates the image file that you want patched. The specification must contain the file name; the remaining fields (device, directory, file type, and version) can be omitted. PATCH uses your default device and directory, assumes a file type of EXE, and uses the highest version of the specified file.

The following example shows one way of invoking PATCH for an interactive PATCH session:

```
$ PATCH DB1:[MASSEY]CIRCLE
```

In this example, PATCH is invoked to apply one or more patches to the image file CIRCLE.EXE. The image file is located on DB1:[MASSEY] and because the version field is omitted, PATCH updates the highest version of CIRCLE.EXE. After you press the RETURN key, PATCH displays the following introductory message:

```
PATCH VERSION 2.0      15 MAR 80

PATCH>
```

The PATCH prompt indicates that PATCH is ready to accept input.

3.1.2 Submitting a Command Procedure for Interactive Execution

When you invoke PATCH to process a command procedure interactively, the file specification indicates the command procedure that contains:

- The name of the image file that is to incorporate the patches
- The PATCH commands that, when processed, perform the necessary changes to the image file

The at sign (@) preceding the file specification tells PATCH that the file is a command procedure. The specification must contain the file name; the remaining fields (device, directory, file type, and version) can be omitted. PATCH uses your default device and directory, assumes a file type of COM, and uses the highest version of the specified file.

The following example illustrates how to process a command procedure interactively. Read Section 2.4 for information on how to create a command procedure.

```
$ PATCH @UPDATE.COM;3

PATCH VERSION 2.0 15 MAR 80

AVERAGE+6:  DIVL2 R5,(R8)
AVERAGE+9:  CLRW (R11)
old:  AVERAGE+9:  CLRW (R11)
new:  AVERAGE+9:  CLRL (R11)
%PATCH-I-WRTFIL, updating image file DB1:  [JAMES.FORT PROGS] AVERAG.EXE;10
$
```

In this example, the command procedure UPDATE.COM is submitted for PATCH processing. The display indicates the patch and the file to which the patch is applied. The DCL prompt ($) indicates completion of the command procedure.

You can also process selected patches in a command procedure. To do so, specify the /UPDATE command qualifier with the PATCH command. Read Section 3.2.2 for complete information, with examples, on this topic.

3.1.3 Submitting a Command Procedure for Batch Execution

Use the third method when you want to submit a command procedure for batch execution. When you do so, the file specification indicates the name of the command procedure that contains:

- The PATCH command which invokes PATCH
- The name of the image file that is to incorporate the patches
- The PATCH commands that, when processed, perform the necessary changes to the image file.

The SUBMIT command enters the command procedure in the batch job queue. You must specify the file name of the command procedure. The remaining fields (device, directory, file type, and version) can be omitted. The SUBMIT command uses your default device and directory, assumes a file type of COM, and uses the highest version of the specified file.

See the VAX/VMS Guide to Using Command Procedures for detailed information and reference material on submitting files for batch execution.

3.2 PATCH COMMAND QUALIFIERS

The command qualifiers PATCH recognizes are /JOURNAL, /OUTPUT, and /UPDATE. These qualifiers are optional, but when used let you:

- Override default file specifications
- Process selected patches in a command procedure

You can enter more than one command qualifier on a single line; however, each qualifier must be prefixed with the slash character (/).

Table 3-1 lists and describes these qualifiers briefly. The sections following the table provide more detailed descriptions, with examples, of the qualifiers.

Table 3-1
PATCH Command Qualifiers

Format	Description
/JOURNAL [= file-spec]	Overrides the default journal file specification
/OUTPUT [= file-spec]	Overrides the default output image file specification
/UPDATE [= (eco-level [,...])]	Processes only those patches associated with the specified ECO levels

3.2.1 Command Qualifiers for Overriding Default File Specifications

The qualifiers that let you override default file specifications are /JOURNAL and /OUTPUT; the /JOURNAL qualifier lets you change the file specification of the journal file, and the /OUTPUT qualifier lets you change the file specification of the output image file.

You do not have to specify all of the fields in the file specification of the /JOURNAL and /OUTPUT qualifiers. PATCH supplies the default values for the fields that you omit. The default values supplied by PATCH are listed and defined in Table 3-2.

Table 3-2
Default Values for the /JOURNAL and /OUTPUT Command Qualifiers

Command Qualifier	Field	Default Value
/JOURNAL [= file-spec]	Device and directory	Process's current defaults
	File name	Input image file name
	File type	JNL
	Version number	1
/OUTPUT [= file-spec]	Device and directory	Process's current defaults
	File name	Input image file name
	File type	EXE
	Version number	1 greater than the highest version of the input image file

The following examples show some valid formats for invoking PATCH with the /JOURNAL and /OUTPUT command qualifiers:

```
$ PATCH AVERAGE /JOURNAL = DB1:[JACKSON]TEST
```

This command invokes PATCH for an interactive PATCH session with the image file AVERAGE.EXE. The /JOURNAL qualifier requests that the journal file be named TEST.JNL and that it be created in the DB1:[JACKSON] directory.

```
$ PATCH PAYROLL /JOURNAL = TESTER /OUTPUT = TESTER
```

This command invokes PATCH for an interactive PATCH session with the image file PAYROLL.EXE. The journal file and the output image file created by this session are named TESTER.JNL and TESTER.EXE, respectively and reside in the current default device and directory.

3.2.2 Command Qualifier for Processing Selected Patches

The /UPDATE command qualifier lets you apply to an image file only the patches that correspond to the ECO levels specified with /UPDATE. When you use /UPDATE, the file specification can denote either a command procedure that contains the patches you want processed or an image file to which certain patches are applied. The formats for the use of the /UPDATE qualifier with the PATCH command are given below.

For command procedures, use:

```
$ PATCH /UPDATE = (eco-level [,...]) @file-spec
```

For image files, use:

```
$ PATCH /UPDATE = (eco-level [,...]) file-spec
```

When you use /UPDATE, specify at least one ECO level. If you specify more than one ECO level, separate the ECO levels with commas, and enclose the entire list within parentheses.

3.2.2.1 **Processing Selected Patches in Command Procedures** - PATCH assumes that the file is a command procedure if the file specification is preceded by an at sign (@). PATCH reads the command procedure and executes those patches corresponding to the ECO levels specified with the /UPDATE qualifier. The first entry in the command procedure is always the name of the image file to which the patches are to be applied.

As mentioned in the section that describes creating command procedures (Section 2.4.2), if a command procedure contains symbolic references, you should not try to apply selected patches from that command procedure to an image file. This is because PATCH parses each patch in the command procedure to determine if it is represented by an ECO level specified with the /UPDATE command qualifier. During the parse operation, if PATCH encounters a symbol defined in a patch that is not to be applied to the image file, it reports an error in the command procedure.

When PATCH reads the command procedure, it displays on the terminal the following information:

1. The PATCH introductory message
2. Patches applied to the image file
3. ECO error messages
    a. ECO levels specified with /UPDATE, but already applied to the image file
    b. ECO levels present in the command procedure, but not specified with /UPDATE

The following example shows how PATCH executes selected patches in the command procedure TEST1.COM:

```
$ PATCH /UPDATE = (10,12) @[JOHNSON.FORTRAN]TEST1.COM

  PATCH   VERSION 2.0   15 MAR 80

DRM1:  0EFDE4800
old:  DRM1:  0EFDE4800
new:  DRM1:  00000000
%PATCH-I-WRTFIL, updating image file DB1:[MATTHEWS.FORTRAN]PROGB.EXE;8
%PATCH-I-UPDATE, patch with eco 11 ignored due to update qualifier
AMSTR+50:  6B5C4005
old:  AMSTR+50:  6B504005
new:  AMSTR+50:  00000000
%PATCH-I-WRTFIL, updating image file DB1:[MATTHEWS.FORTRAN]PROGB.EXE;8

$
```

This PATCH command executes the command procedure [JOHNSON.FORTRAN] TEST1.COM in interactive mode. The /UPDATE qualifier requests that only the patches identified by the ECO levels 10 and 12, contained in TEST1.COM, be applied to the image file PROGB.EXE in the directory [MATTHEWS.FORTRAN]. The display indicates that the patches specified with /UPDATE were successfully applied to PROGB.EXE and reports that the patch identified by ECO level 11 was not applied because that ECO level was not specified with the /UPDATE qualifier.

Note that when you execute command procedures interactively, the /UPDATE qualifier must precede the name of the command procedures.

If you do not include the /UPDATE qualifier, PATCH processes all patches in the command procedure.

3.2.2.2 **Processing Selected Patches Interactively** - When you include the /UPDATE qualifier with the input image file, PATCH processes only those patches represented by the ECO levels specified with /UPDATE. If, while editing the image file, you define an ECO level not specified with the /UPDATE qualifier, PATCH ignores the subsequent commands and displays an informational error message. For example:

```
$ PATCH /UPDATE = (2,4) CIRCLE

  PATCH  VERSION 2.0   15 MAR 80

PATCH> SET ECO 2
    .
    .
    .
PATCH>UPDATE
    .
    .
    .
PATCH> SET ECO 3
%PATCH-I-UPDATE, patch with eco level 3 ignored due to update qualifier
PATCH> SET ECO 4
```

In the above example, PATCH allows processing of the patch identified by the ECO level 2; however, when you try to define ECO level 3, PATCH displays a message indicating that the patch cannot be applied because it was not specified with /UPDATE.

Note also that if you specify an ECO level with /UPDATE, but do not set that ECO level during the PATCH session, PATCH issues an informational error message when you exit from PATCH.

3.3 HOW TO TERMINATE PATCH

To terminate a PATCH session, type the EXIT command. This command passes control back to the command interpreter. Keep in mind, though, that to create an output image file, you must type the UPDATE command before exiting from PATCH. For example:

```
PATCH> UPDATE
%PATCH-I-WRTFIL, updating image file DB2:[DARNELL]BINDER.EXE;10
PATCH> EXIT
$
```

If you omit the UPDATE command before you type EXIT, control is passed to the command interpreter, but the output image file is not written. The commands are still recorded in the journal file.

To interrupt program execution, type CTRL/Y (echoed as ^Y). This command also passes control back to the command interpreter. To continue patching, type the CONTINUE command. For example:

```
PATCH> ^Y
$ SHOW TIME
  3-AUG-1979 10:35:53
$ CONTINUE
PATCH>
```

You can also terminate PATCH by typing CTRL/Z (echoed as ^Z). This is equivalent to typing EXIT.

CHAPTER 4

SYMBOLS

PATCH is a symbolic utility. This means that you can pass symbol information to PATCH or define new symbols during a PATCH session, then use that symbolic information as a reference tool. The symbol information available to PATCH is determined by certain qualifiers you include or exclude when compiling or assembling and linking a program.

4.1 SYMBOLS RECOGNIZED BY PATCH

PATCH recognizes the following symbol types:

- Global symbols
- Local symbols
- Module names, program section names, and routine names
- Universal symbols
- Symbolic instruction labels
- Symbols defined with the PATCH DEFINE command

PATCH also recognizes patch area symbols. This symbol type is described in Chapter 5.

PATCH cannot access local labels in VAX-11 MACRO programs. Local labels are symbols that consist of a number (0 through 65535) followed by the dollar sign ($).

4.1.1 Global Symbols

Global symbols are symbols defined in one module to be accessed by any number of other modules.

You indicate a global symbol in your source program by delimiting it with a special operator. This operator depends on the language in which your program is written. For example, in VAX-11 MACRO, you define a global symbol by using double colons (::) or equal signs (==). (Individual language reference manuals contain information on defining global symbols.)

To pass global symbol information to PATCH, specify the /DEBUG qualifier when you link your program. (See the appropriate language manual for complete information on passing global symbols.)

4.1.2 Local Symbols

Local symbols are symbols that are defined in a particular module and can be accessed only by that module.

You indicate a local symbol in your source program by delimiting it with a special operator. This operator depends on the language in which your program is written. For example, in VAX-11 MACRO, you define a local symbol by using a single colon or equal sign. (Individual language reference manuals contain information on defining local symbols.)

To pass local symbol information to PATCH, you must include the debugger when you assemble or compile and link your program. (See the appropriate language manual and the VAX-11 Linker Reference Manual for more information on passing local symbols.) When you run PATCH, use the SET MODULE command to enter local symbol information into the symbol table.

4.1.3 Module Names, Program Section Names, and Routine Names

Module names, program section names, and routine names are always passed to PATCH unless you specify /NODEBUG and /NOTRACEBACK at link time. When these qualifiers are specified, no symbol information is passed to PATCH.

To limit symbol information to only module names, program section names, and routine names, link your program with the /TRACEBACK and /NODEBUG qualifiers (these are the default qualifiers).

4.1.4 Universal Symbols

Universal symbols are:

- A subset of the global symbols of a shareable image
- The only type of symbol PATCH can reference when patching a shareable image

Any global symbol is eligible to be a universal symbol. However, you typically declare only a limited subset of global symbols to be universal, because not all symbols are required for a particular application and partial symbol declaration reduces system overhead.

To declare a symbol to be universal, specify the option UNIVERSAL= in the options file when you link your program. For example:

```
$ LINK/SHAREABLE PROGA,PROGB/OPTIONS
```

PROGB.OPT

```
.
.
.
UNIVERSAL=A,B,C
.
.
.
```

In the above example, the symbols A, B, and C are declared universal in the shareable image PROGA. The file named PROGB.OPT is the linker options file in which the universal symbol declarations are contained.

To reference universal symbols in a shareable image, observe the following guidelines:

- If the shareable image is position independent, all relocatable universal symbols are based at zero (not at hexadecimal 200, as the linker map may suggest). Therefore, when you evaluate a universal symbol, add 200 to the value displayed by the EVALUATE command.

- If a symbol is in a transfer vector and has been declared universal by use of the .TRANSFER directive, the value of the symbol in the shareable image's user symbol table (UST) is the address of the transfer vector, not the actual routine address in the image.

Thus, for PATCH to reference locations symbolically in a non-zero-based shareable image that is position independent, you must specify the symbol as:

```
symbol + <base>
```

The following example illustrates the proper use of universal symbols for patching a shareable image.

```
$ MACRO CHAL1
$ LINK/SHAREABLE CHAL1,X/OPTIONS
UNIVERSAL=LBR$BENN
$ PATCH CHAL1

  PATCH VERSION 2.0 15 MAR 80

%PATCH-I-NOLCL, image does not contain local symbols
PATCH> EXAMINE LBR$BENN
%PATCH-E-NSADDR, address 00000050 is not within image
PATCH> EVALUATE LBR$BENN
00000050
PATCH> EXAMINE/WORD 250
00000250:   0FFC
PATCH> EXAMINE/INSTRUCTION
00000252:   BRW     000008EE
PATCH> EXAMINE/INSTRUCTION #8EE
000008EE:  SUBL2  #08,SP
```

In this example, the first EXAMINE command tries unsuccessfully to display the contents of the universal symbol LBR$BENN. The EVALUATE command verifies that LBR$BENN is memory location 50. Because the shareable image is position independent, you must add the value 200 to location 50 and the word of data in the resulting location (location 250) is displayed; this data represents the entry point register save mask of the routine LBR$BENN. To display the first instruction in LBR$BENN, the EXAMINE/INSTRUCTION command is used. In this case, the branch instruction displayed is the transfer to the actual start of LBR$BENN. Examining the destination of the branch instruction provides the first instruction in LBR$BENN.

4.1.5 Symbolic Instruction Labels

Symbolic instruction labels are 1- to 15-character symbols having the following characteristics:

- They must start with an alphabetic character (labels such as 10$ are invalid for use with PATCH)
- They can consist of alphanumeric characters, dollar signs ($), and/or underscores (_)

A symbolic instruction label lets you associate a specific name with a particular location without knowing the numeric address of the location. This situation generally arises when:

1. A series of instructions is inserted, replaced, or deposited into the existing code (See Chapter 8)
2. PATCH places the new instructions in the current patch area and generates branch instructions to and from the patch area to maintain the logical flow of program execution (See Chapter 5)
3. Of the new instructions being added, one of the instructions references another one

Because you do not know the exact location of the instruction being referenced, you can use a symbolic instruction label to reference the location. For example:

```
PATCH> INSERT/INSTRUCTION 350
OLD> 'TSTL R3'
OLD> EXIT
NEW> 'BEQL NONE'
NEW> 'MOVL R4,R0'
NEW> 'DIVL2 R0,R1'
NEW> 'MULL2 R2,R1'
NEW> 'NONE:  RET'
NEW> EXIT
```

Here, a group of instructions is inserted after location 350 by means of a patch area. Note that the symbolic instruction label NONE is used in the first instruction to reference the fifth instruction. The label NONE is used because the exact placement of these instructions is not known.

Side Effect of Using Symbolic Instruction Labels: A side effect of using symbolic instruction labels is that once a symbolic instruction label is assigned a location, the label always references that location, even if the contents of the location are moved. For example, suppose you have the following code in your program:

TEST.EXE

```
  .                          .
  .                          .
  .                          .
500                  TSTL     R3
502          XX:     BEQL     LBL3
505          YY:     CLRL     -(R6)
  .                          .
  .                          .
  .                          .
```

```
550                     TSTL    R5
552         LBL3:       MOVL    R2,R3
 .                        .
 .                        .
 .                        .
```

In this situation, the symbolic instruction label LBL3 is assigned to location 552, and a preceding instruction (BEQL LBL3) references LBL3. Later, you discover that the TSTL R5 instruction is wrong and perform the following patch:

```
PATCH> REPLACE/INSTRUCTION 550
OLD> 'TSTL R5'
OLD> EXIT
NEW> 'CMPL R5,R6'
NEW> EXIT
old:    00000550:   TSTL R5
old:    LBL3:   MOVL R2,R3
new:    00000550:    BRW 00007800
new:    00000553:    NOP
new:    00000554:    NOP
new:    00007800:    CMPL R5,R6
new:    00007803:    MOVL R2,R3
new:    00007806:    BRW 00000555
```

During the replacement operation, the following occur:

- The CMPL R5,R6 instruction requires more bytes than the TSTL R5 instruction, thus preventing a simple replacement from being performed.
- The CMPL R5,R6 instruction is moved to the current patch area and a branch to the patch area replaces the TSTL R5 instruction.
- The branch instruction needed to reroute the logical flow of execution to the patch area also requires more bytes than those occupied by the TSTL R5 instruction.
- The instruction that follows the TSTL R5 instruction is also moved to the patch area to make room for the branch instruction. (This is the MOVL R2,R3 instruction, which has the symbolic instruction label LBL3 assigned to its location.)
- The branch instruction is inserted into the area left blank by the deletion of the TSTL R5 instruction and removal of the MOVL R2,R3 instruction to the patch area.
- NOP instructions are used to fill the locations not used by the branch instruction.

As a result of this patch, the symbolic instruction label LBL3 still references location 552, which is the middle of the branch instruction. If you were to examine LBL3, the following instruction would be displayed:

```
PATCH> EXAMINE/INSTRUCTION LBL3
LBL3:   MNEGD #01,#01
```

To remedy the situation, modify all instructions that reference LBL3, as demonstrated below:

```
PATCH> REPLACE/INSTRUCTION XX
OLD> 'BEQL LBL3'
OLD>'CLRL -(R6)'
OLD> EXIT
NEW> 'BNEQ LBL2'
NEW> 'BRW 7803'
NEW> 'LBL2: CLRL -(R6)'
NEW> EXIT
```

To correct the reference made to the MOVL R2,R3 instruction and to maintain a logical flow of program execution, the BEQL LBL3 instruction and CLRL -(R6) were replaced with complementary instructions:

Instruction	Description
BNEQ LBL2	branch if not equal to label LBL2
BRW 7803	unconditional branch to location 7803

Note that the unconditional branch instruction uses a word displacement to ensure that the branch can reach its destination.

4.1.6 Symbols Defined with the DEFINE Command

When you run PATCH, you can assign a symbolic name to a location or value by using the DEFINE command. This command lets you supplement or override existing symbols in your image for the duration of the PATCH session.

See the DEFINE command in Chapter 8 for more details about this command.

4.2 THE PATCH SYMBOL TABLE

When PATCH starts up, it builds a symbol table in which global symbols, local symbols, universal symbols, module names, program section names, routine names, patch area symbols, and symbols defined with the DEFINE command are placed. Global and local symbols are placed in the symbol table if you followed the rules for passing these symbols. If you did not pass these symbols, PATCH displays one or both of the following informational error messages when it is invoked:

```
%PATCH-I-NOGBL, some or all global symbols are not accessible
%PATCH-I-NOLCL, image file does not contain local symbols
```

PATCH uses symbol table information to translate symbols into values and values into symbols. The rules governing symbol and value translations are discussed in Section 4.4.

PATCH has no knowledge of symbol information that is not present in the symbol table. To confirm that the symbol table contains the appropriate entries use the SHOW MODULE command. This command reports the current symbol table status. For more information on the SHOW MODULE command, see Chapter 8.

4.3 PATHNAMES

Some symbols may be multiply defined within your image file. To find a particular location that is defined by a multiply defined symbol, you must create an unambiguous path that directs PATCH to the location you want to reference.

A pathname is a symbolic expression that uniquely identifies a location within your image file. A pathname consists of the following elements:

scope\symbol-name ± offset-value

scope

Identifies a particular module within your image file in which PATCH searches for the symbolic name supplied. The backslash character (\) is a required delimiter.

Supplying a scope is unnecessary when: (1) all the symbol names within your file are unique; (2) the scope is already set, by use of the SET SCOPE command, to the region you want to reference; or (3) the symbol being accessed is a global symbol.

symbol-name

Identifies a particular location within your image file. Symbol names can be global symbols, local symbols, or symbols you defined with the DEFINE command.

The symbol name is the only element that must always be supplied in order to find that location.

± offset-value

Specifies a positive or negative numeric value, in the current radix, appended to the symbol name. The offset value is used to reference unlabeled locations.

The following example demonstrates the use of a pathname:

```
PATCH>EXAMINE/INSTRUCTION REM_MOD1\CRM_STRT+1A
REM_MOD1\CRM_STRT+1A:  CLRQ R0
```

In this example, the EXAMINE/INSTRUCTION command requests that the contents of location CRM_STRT+1A, in the module named REM_MOD1, be displayed as a VAX-11 MACRO instruction.

4.4 TRANSLATING SYMBOLS AND VALUES

PATCH follows a specific sequence of steps to translate an address value to a symbol or a symbol to an address value. The following sections explain the algorithms PATCH uses to compute these translations.

4.4.1 Translating Symbols into Address Values

PATCH's translation of symbolic entries into values is controlled by the GLOBALS-NOGLOBALS and SCOPE-NOSCOPE modes. The following steps outline the procedure PATCH uses to perform this translation.

1. First, PATCH compares the symbolic entry with all user-defined symbols for an exact match.

2. If step 1 fails, PATCH checks the status of the GLOBALS-NOGLOBALS mode setting. If the mode is set to GLOBALS, PATCH compares the symbolic entry as you entered it with the symbols in the symbol table, looking for an exact match. That is, when PATCH searches the symbol table for a match, it does not prefix the contents of the scope to the symbolic entry.

3. If step 2 fails, or if the mode is not set to GLOBALS, PATCH checks the status of the SCOPE-NOSCOPE mode setting. If the mode is set to SCOPE, PATCH prefixes the contents of the scope setting to the symbolic entry and searches the symbol table for an exact match.

4. If step 3 fails, or if the mode is not set to SCOPE, PATCH prefixes the contents of the scope (based on the current PC) to the symbolic entry and searches the symbol table for an exact match.

5. If step 4 fails, PATCH again checks the status of the GLOBALS-NOGLOBALS mode setting. If the mode is set to NOGLOBALS, PATCH compares the symbolic entry as you entered it with the symbols in the symbol table, looking for an exact match. That is, when PATCH searches the symbol table for a match, it does not prefix the contents of the scope setting to the symbolic entry. (Note that this is the same procedure as step 2, except the mode is set to NOGLOBALS.)

If the translation is successful, PATCH evaluates the expression in which the symbolic entry appears. If the translation fails, PATCH issues the following error message to indicate that it could not evaluate the symbolic entry:

```
%PATCH-W-NOSYMBOL, no such symbol 'symbolic-entry'
```

4.4.2 Translating Address Values into Symbols

PATCH's translation of address values into symbolic entries is governed by the SYMBOLS-NOSYMBOLS mode. The following steps outline the procedure PATCH uses to perform this translation:

1. First PATCH compares the address value with all user-defined symbols for an exact match.

2. If step 1 fails, PATCH compares the address value with the global and local symbol definitions, looking for an exact match.

3. If step 2 fails, PATCH searches all symbol definitions, looking for the one whose corresponding address value is closest to, but less than, the specified value. PATCH rejects a symbol definition as being closest to the address value when the difference between the symbol and the address value is greater than or equal to hexadecimal 100.

If the translation is successful, PATCH evaluates the expression in which the address value appears and reports the address value in terms of the corresponding symbol name. If the translation fails, PATCH still evaluates the expression in which the address value appears; however, PATCH reports the address value as a numeric address in terms of the current radix.

CHAPTER 5

PATCH AREA

Patch area is read/write memory located within or at the end of an image file used to store additional instructions or data entries. Default patch area, that is, patch area created by PATCH, is always appended to the end of an image file. User-defined patch area, on the other hand, can be located anywhere within the image file.

This chapter discusses the two types of patch area, the patch area string descriptor used to keep track of the status of the patch area, the patch area symbols generated by PATCH, and the commands that affect patch area.

5.1 DEFAULT PATCH AREA

The default patch area is area automatically supplied by PATCH. Generally, the default patch area is expandable. It is limited by available disk space during PATCH execution and the value of the SYSGEN virtual page count parameter at run time.

PATCH allocates the default patch area one page at a time, as necessary. For example, if you are inserting additional instructions into the image file, PATCH allocates another page of the default patch area if the existing default patch area is too small to accommodate the new instructions.

By convention, whenever you patch an image file and the patches require the use of a patch area, those patches are placed in the default patch area. However, in some instances, the default patch area cannot be used to store patches. Such is the case for device driver images and shareable images that have been linked with other images.

If you need to patch a device-driver image or a shareable image that has been linked with other images, you should use a user-defined patch area as described in the following section.

5.2 USER-DEFINED PATCH AREA

A user-defined patch area is area you create in the source code. This patch area is located within the boundaries of the image file; PATCH does not extend the length of the image file to accommodate a user-defined patch area. Also, a user-defined patch area is of a fixed size; PATCH cannot increase the size of a user-defined patch area to accommodate additional patches.

You should use a user-defined patch area to store additional code when patching device-driver images and shareable images. Section 5.2.1 below explains why a user-defined patch area is preferred in these cases over the default patch area. Sections 5.2.2 through 5.2.4 then describe how to create a user-defined patch area, access it, and terminate the use of it.

5.2.1 When to Use User-Defined Patch Area

As mentioned above, you should use a user-defined patch area rather than the default patch area when patching an image whose total size cannot be increased. Examples of this type of image are device-driver images and shareable images.

A device-driver image contains, at the start of the image, a location that defines the total length of the driver. When the device driver is loaded, the driver loader looks at this location to determine the amount of space to allocate to the device-driver. The fact that PATCH extends the device-driver image when default patch area is used means that the total length of the device-driver image increases. However, PATCH does not recognize the location that contains the length of the device-driver, nor can PATCH modify the contents of that location if it extends the device-driver image. Thus, a subsequent running of a device driver that uses default patch area causes the patch area to be truncated.

Similarly, a program linked with shareable images must know, at link time, the amount of address space (in pages) occupied by each shareable image. When the program is linked, it leaves virtual address space in the image file in which the shareable images will be placed. If the length of a shareable image should grow after the program is linked, then at run-time, the patch area will overlay part of the shareable image.

5.2.2 Creating User-Defined Patch Area

To create a user-defined patch area, set aside area at assembly or compile time. The following example illustrates how to set aside patch area in a VAX-11 MACRO program. The numbers to the right of the example are keyed to the descriptions below.

```
PATSIZE    =  size-in-bytes-of-patch-area  (1)
PATDESC::
     .LONG     PATSIZE   (2)
     .LONG     PATADR
PATADR::
     .BLKB     PATSIZE   (3)
```

In the preceding example, you:

(1) Define a size for the patch area

(2) Create a patch area descriptor that specifies the size and first free byte of the patch area

(3) Define a symbol to represent the starting address of the patch area

5.2.3 Accessing User-Defined Patch Area

After creating a user-defined patch area, you can access that patch area by issuing the SET PATCH_AREA command. This command is fully described in Chapter 8.

5.2.4 Terminating the Use of User-Defined Patch Area

To terminate the use of a user-defined patch area, issue the CANCEL PATCH_AREA command. This command resets the current patch area to the default patch area (See Chapter 8).

You can also reset the patch area from one user-defined patch area to another user-defined patch area by issuing the SET PATCH_AREA command. (This command is fully described in Chapter 8.)

5.3 PATCH AREA DESCRIPTOR

A patch area descriptor is a record of the current status of a patch area. The first longword contains the current size (in unused bytes) of the patch area, and the second longword contains the address of the first free byte of the patch area. You can observe the status of the current patch area by issuing the SHOW PATCH_AREA command. For example:

```
PATCH> SHOW PATCH_AREA
current patch area size:      00000200
current patch area address:   00007800
```

All patch area, whether it is user-defined or default patch area, has a descriptor associated with it. Any time a command is executed that affects patch area, the patch area descriptor is updated to reflect the modifications.

5.4 PATCH AREA SYMBOLS

DIGITAL reserves the use of certain PATCH symbols to represent the start of particular patch areas. These symbols range from PAA through PZZ and are referred to as patch area symbols. You are not recommended to use these symbols. However, you can create your own patch area symbols using the ALIGN command. This command lets you assign symbolic names to the starting address of the default patch area and to all the user-defined patch areas. This command is fully described in Chapter 8.

5.5 COMMANDS THAT AFFECT PATCH AREA

The commands that affect patch area are INSERT/INSTRUCTION, REPLACE/INSTRUCTION, DEPOSIT/PATCH_AREA, and ALIGN. The following four sections briefly describe how these commands affect the patch area.

5.5.1 INSERT/INSTRUCTION Command

When you use the INSERT/INSTRUCTION command, PATCH automatically stores the new instructions in patch area and generates unconditional branch instructions to reroute the logical flow of execution to the patch area, then back to the inline code.

Occasionally, when you insert one or more new instructions into your code, the following events may occur:

- A branch-type instruction is moved to the current patch area
- The patch area is far enough away to prevent the branch-type instruction from reaching its destination

When these events occur, the INSERT/INSTRUCTION command recalculates the relative displacement of the branch-type instruction to allow the branch to reach its destination. For example, suppose the insertion of a new instruction causes the BRW 200 instruction to be moved to patch area. However, once the BRW 200 instruction is in the patch area, the word displacement is not large enough to reach location 200. The INSERT/INSTRUCTION command then recalculates the word displacement to allow the branch to succeed.

Be aware that PATCH always completes a patch with a branch instruction to the inline code, even if the last instruction of the patch is an unconditional branch instruction.

See Chapter 8 for more information on the INSERT command.

5.5.2 REPLACE/INSTRUCTION Command

When you use the REPLACE/INSTRUCTION command, PATCH automatically stores the new instructions in patch area only if the new instructions occupy more bytes in memory than the instructions being replaced. If patch area is used, PATCH generates branch instructions to and from the patch area to maintain the correct flow of program execution.

As with the INSERT/INSTRUCTION, the REPLACE/INSTRUCTION command can also cause a branch-type instruction to be moved to the patch area. If the new location of the branch-type instruction is too far away to allow the branch to reach its destination, the REPLACE/INSTRUCTION command recalculates the relative displacement of the branch-type instruction to allow the branch to reach its destination. (Read the previous section to see exactly when and how branch-type instructions are modified.)

When you replace numeric or ASCII data, the new data is truncated if it exceeds the length of the data that it is to replace. (That is, patch area is not used.)

Be aware that PATCH always completes a patch with a branch instruction to the inline code, even if the last instruction of the patch is an unconditional branch instruction.

See Chapter 8 for more information on the REPLACE command.

5.5.3 DEPOSIT/PATCH_AREA Command

The DEPOSIT/PATCH_AREA command uses patch area to store additional instructions or data; however, PATCH does not automatically generate branch instructions to and from the patch area when you deposit the new data. You must insert the branch instructions manually. You must also recalculate the relative displacements of all branch-type instructions affected by the insertion of new data or instructions. See Chapter 8 for more information on the DEPOSIT command.

5.5.4 ALIGN Command

Use the ALIGN command to allocate space for the default patch area in an image file that has not previously been patched or to redefine the first free byte of the current patch area. See Chapter 8 for more information on the ALIGN command.

CHAPTER 6

ENTRY AND DISPLAY MODES

Entry and display modes determine how PATCH interprets your commands and displays its output. The entry and display modes are grouped into four functional categories: context, length, radix, and symbol search. The context modes control whether addresses can be represented symbolically, whether instructions can be represented as VAX-11 MACRO instructions, and whether data can be represented as ASCII text. The length modes determine the length in which data entries are represented. The radix modes determine the base in which addresses and data entries are represented. The symbol search modes control how PATCH translates symbols into addresses and vice versa.

Table 6-1 defines the purpose of each mode. Sections 6.1 and 6.2, which follow the table, describe the techniques for establishing a particular mode setting and the rules to follow when specifying the entry and display modes.

Table 6-1
Entry and Display Modes in PATCH

Category	Mode	Description	Initial Default
Context	INSTRUCTION	Accepts or displays your data as VAX-11 MACRO instructions. If you enter old or new data, and this mode is set, the instructions must be enclosed within apostrophes or quotation marks.	NOINSTRUCTION
	NOINSTRUCTION	Inhibits the entry or display of data as VAX-11 MACRO instructions. This is the initial default condition.	
	ASCII	Accepts or displays your data as ASCII characters. If you enter old or new data, and this mode is set, the data must be enclosed within apostrophes or quotation marks.	NOASCII
	NOASCII	Inhibits the entry or display of your data as ASCII characters. This is the initial default condition.	
	NOSYMBOLS	Inhibits the display of addresses symbolically.	SYMBOLS
	SYMBOLS	Displays addresses symbolically. This is the initial default condition.	

(continued on next page)

Table 6-1 (Cont.)
Entry and Display Modes in PATCH

Category	Mode	Description	Initial Default
Length	BYTE	Accepts or displays a byte of data	LONG
	WORD	Accepts or displays a word of data	
	LONG	Accepts or displays data in longword lengths. This is the initial default condition.	
Radix	OCTAL	Accepts or displays data and addresses in octal representation.	HEXADECIMAL
	DECIMAL	Accepts or displays data and addresses in decimal representation.	
	HEXADECIMAL	Accepts or displays data and addresses in hexadecimal representation. This is the initial default condition.	
Symbol Search	NOSCOPE	Inhibits using SCOPE's contribution to a pathname, and reverts to <null> scope.	SCOPE
	SCOPE	Allows using SCOPE's contribution to a pathname. This is the initial default condition.	
	GLOBALS	Allows the use of the symbolic entry as the *first* pathname in a search.	NOGLOBALS
	NOGLOBALS	Allows the use of the symbolic entry as the *last* pathname in a search. This is the initial default condition.	

6.1 MODES AND MODE QUALIFIERS

You can modify the mode settings with the SET MODE and CANCEL MODE commands or by issuing an alternative mode qualifier on a command line. For example:

```
PATCH> SET MODE INSTRUCTION
PATCH> DELETE 400='MOVL (R6),(R5)'
```

or

```
PATCH> DELETE/INSTRUCTION 400='MOVL (R6),(R5)'
```

The above examples are roughly equivalent. The only difference is that in the first example INSTRUCTION mode becomes a default setting, whereas in the second example INSTRUCTION mode is a temporary setting effective only for that command line. The next command line adheres to the current mode settings.

To examine the status of the current modes, issue the SHOW MODE command.

6.2 SPECIFYING ENTRY AND DISPLAY MODES

This section discusses the rules for specifying the entry and display modes. Entry and display modes control the format of your patches. For example, to deposit a VAX-11 MACRO instruction, the INSTRUCTION mode must be set, whereas to deposit some ASCII text, the ASCII mode must be set.

6.2.1 Context Modes

The context modes are:

INSTRUCTION	NOINSTRUCTION
ASCII	NOASCII
SYMBOLS	NOSYMBOLS

These modes serve as ON-OFF switches. For example, INSTRUCTION mode attempts to display the contents of one or more locations as VAX-11 MACRO instructions. NOINSTRUCTION mode then inhibits the ability to display the contents of one or more locations as VAX-11 MACRO instructions.

The following subsections describe each of the context modes, including the function of each mode and the rules PATCH follows when a particular mode is set.

6.2.1.1 **INSTRUCTION-NOINSTRUCTION Modes** - The INSTRUCTION-NOINSTRUCTION modes control whether data can be entered and displayed as VAX-11 MACRO instructions.

When INSTRUCTION mode is set, PATCH:

- Accepts and displays data as VAX-11 MACRO instructions.
- Ignores the current length mode when accepting or displaying instructions. PATCH increments the current address by the number of bytes occupied by the instruction.

- Invokes, when you use the REPLACE, INSERT, or DEPOSIT/PATCH_AREA command, an automatic fitting mechanism that takes symbolic instructions and fits them into the locations indicated. This fitting mechanism may include moving some instructions into the current patch area and inserting branches to them, or replacing unused bytes with no operation instructions (NOPs).

- Requires that all symbolic instructions entered as input must be enclosed within quotation marks or apostrophes. (See "Entering VAX-11 MACRO Instructions" in Section 7.4 for more information.)

When NOINSTRUCTION mode is set, PATCH accepts and displays data according to the current radix, length, and ASCII-NOASCII mode setting.

6.2.1.2 **ASCII-NOASCII Modes** - The ASCII-NOASCII modes control whether data can be entered and displayed as ASCII data.

When INSTRUCTION mode is set, PATCH ignores the ASCII-NOASCII setting and accepts and displays data as VAX-11 MACRO instructions.

When ASCII and NOINSTRUCTION modes are set, PATCH:

- Requires that all ASCII data entered as input be enclosed within quotation marks or apostrophes. (See "Entering ASCII Data strings" in Section 7.3 for more information.)

- Displays an error message if you enter a string of numbers not enclosed within quotation marks or apostrophes.

- Truncates an ASCII input string if the string exceeds the limit imposed on it by the current length mode. PATCH truncates input strings by discarding excessive characters from the right.

- Displays ASCII output strings according to the current length mode. For example:

```
PATCH> EXAMINE/BYTE/ASCII 675
00000675:     I
PATCH> EXAMINE/WORD/ASCII 675
00000675:     ID
PATCH> EXAMINE/LONG/ASCII 675
00000675:     IDLE
```

- Ignores the current radix mode when accepting or displaying data.

When NOASCII and NOINSTRUCTION modes are set, PATCH accepts and displays data according to the current radix and length mode settings.

6.2.1.3 **SYMBOLS-NOSYMBOLS Modes** - The SYMBOLS-NOSYMBOLS modes control whether addresses are displayed by pathnames or symbols, rather than by numeric addresses. (See Section 4.3 for the definition of a pathname.)

When SYMBOLS mode is set, PATCH:

- Displays all locations by their respective pathnames or symbols
- Accepts pathnames as input

When NOSYMBOLS mode is set, PATCH displays all locations by their numeric addresses. You can still enter a pathname as input to PATCH; however, PATCH displays that location by its numeric address.

Chapter 4 describes the rules PATCH follows when it translates address values into symbols and vice versa.

6.2.2 Length Modes

The modes that specify length are BYTE, WORD, and LONG. They denote how many bytes (1, 2, or 4) in memory are available to store a given data item. The following rules apply to the length modes:

- VAX-11 PATCH ignores the current length mode and accepts or displays instructions in their entirety. (This condition exists only when INSTRUCTION mode is set.)
- Truncate the most significant bit positions of numeric data that exceed the boundary imposed on it by the current length mode. (This condition exists only when NOINSTRUCTION and NOASCII modes are set.) PATCH also issues an informational error message. For example:

```
PATCH> DEPOSIT/NOINSTRUCTION/NOASCII/BYTE
LOC> 200
NEW> 123456543
%PATCH-I-NUMTRUNC, number truncated
NEW> EXIT
%PATCH-I-NUMTRUNC, number truncated
old:     00000200:     22
new:     00000200:     43
```

- Set only the last length mode specified when you enter conflicting length modes on a single command line.

6.2.3 Radix Modes

The modes that specify the radix are OCTAL, DECIMAL, and HEXADECIMAL. They refer to the base by which PATCH translates the entry or display of numeric data and addresses. The following rules apply to the radix modes:

- Ignore the radix mode when the context modes are set
- Display data according to the current length mode
- Set only the last radix mode specified when you enter conflicting radix modes on a single command line

You can also change the radix modes by specifying a radix operator (^O,^D,^X,^B). A radix operator controls only the entry that it accompanies; it has no control over the radix in which PATCH displays a value. (See Section 7.9.7, "Radix Operator," for more information.)

6.2.4 Symbol Search Modes

The symbol search modes are:

SCOPE	NOSCOPE
GLOBALS	NOGLOBALS

Like the context modes, the symbol search modes serve as ON-OFF switches. These modes control how PATCH performs symbol translations.

6.2.4.1 **SCOPE-NOSCOPE Modes** - The SCOPE-NOSCOPE modes control whether the contents of the scope setting are prefixed to a symbolic entry when PATCH tries to locate the numeric address represented by that the symbolic entry. Note that PATCH follows a certain procedure when performing symbol-to-value translations. This procedure is based on the current setting of the SCOPE-NOSCOPE and GLOBALS-NOGLOBALS modes. See Chapter 4 for the procedure PATCH uses to perform symbol-to-value translations.

6.2.4.2 **GLOBALS-NOGLOBALS Modes** - The GLOBALS-NOGLOBALS modes, like the SCOPE-NOSCOPE modes, control whether the contents of the scope setting are prefixed to a symbolic entry when PATCH tries to locate the numeric address represented by the symbolic entry. Note that PATCH follows a certain procedure when performing symbol-to-value translations. This procedure is based on the current setting of the SCOPE-NOSCOPE and GLOBALS-NOGLOBALS modes. See Chapter 4 for the procedure PATCH uses to perform symbol-to-value translations.

CHAPTER 7

RULES OF SYNTAX

This chapter provides information on the rules PATCH adheres to when processing commands.

7.1 COMMAND PROMPTING

All PATCH commands, except the EXAMINE command, issue a prompt when insufficient parameters accompany the command name. For example, the DEPOSIT command prompts for a location and the contents you are depositing in that location if you omit these parameters:

```
PATCH> DEPOSIT
LOC> 1363
NEW> 01
```

Some commands require an equal sign (=). When you let PATCH prompt for omitted parameters in such commands, do not specify the equal sign.

Some commands repeatedly prompt for a given parameter when that parameter is omitted. In such cases, type the EXIT command to terminate the prompt. The SET MODE command typifies this rule:

```
PATCH> SET MODE
NEW> INSTRUCTION
NEW> SYMBOLS
NEW> NOGLOBALS
NEW> EXIT
```

The SET MODE command continues to prompt for mode values until you type EXIT to complete the command sequence.

Note that you can enter only one parameter in response to a repetitive prompt. For example, typing:

```
PATCH> SET MODE
NEW> INSTRUCTION, NOASCII, BYTE
```

results in the following error message:

```
%PATCH-W-SYNTAX, command syntax error at or near ',NOASCII,B'
```

The commands CHECK ECO and CHECK NOT ECO accept a single value or a single range of values in response to the repetitive prompt.

7.2 CONTINUING COMMANDS OVER MULTIPLE LINES

You can continue a command over multiple lines by typing a hyphen character (-) as the last nonblank, noncomment character on the command line. For example:

```
PATCH> DEPOSIT -
_> /BYTE-
_> 300 -
_> = 54
```

The _> characters indicate continuation lines.

When you continue commands over more than one line, you must still supply all the required syntax. In the above example, the equal sign (=) must be specified.

7.3 ENTERING ASCII DATA STRINGS

When you enter ASCII data strings to PATCH, they must be enclosed within matching quotation marks (") or apostrophes ('). By allowing either quotation marks or apostrophes to be string delimiters, you can include one or the other delimiter as part of the input string. For example, if you want to insert the ASCII text 'AL', you would type:

```
PATCH> DEPOSIT/ASCII 500 = "'AL'"
```

You cannot delimit an ASCII data string with the same delimiter as the one you include as part of the string. For example, typing the following command results in an error:

```
PATCH> DEPOSIT/ASCII 500 = ''AL''
```

When entering an ASCII data string, PATCH will truncate the string if it exceeds the length imposed on it by the current length mode setting, as shown below:

```
PATCH> SET MODE ASCII,BYTE
PATCH> DEPOSIT 600 = 'ABCDEFG'
%PATCH-I-STGTRUNC, string truncated
%PATCH-I-NUMTRUNC, number truncated
old:     00000600:     ''
new:     00000600:     'A'
```

PATCH truncates the string by discarding excessive characters from the right.

Finally, to enter ASCII data strings, either specify the /ASCII mode qualifier or set ASCII mode. For more information on ASCII mode, see Section 6.2.

7.4 ENTERING VAX-11 MACRO INSTRUCTIONS

The rule for the use of quotation marks (") and apostrophes (') as described for entering ASCII data strings (Section 7.3) also applies for entering VAX-11 MACRO instructions. In brief, an instruction string entry must be delimited by quotation marks or apostrophes and not a combination of both, and the delimiter must not appear within the string.

In addition, to enter VAX-11 MACRO instructions, either specify the /INSTRUCTION mode qualifier or set the INSTRUCTION mode. For more information on INSTRUCTION mode, see Section 6.2.

To enter operands with displacements, you must prefix the operand with B^, W^, or L^. For example:

```
PATCH> DEPOSIT 500 = B^4(R0)
```

7.5 ENTERING NUMERIC DATA

When you enter numeric data to PATCH, the mode qualifiers /NOINSTRUCTION and /NOASCII must be specified or the NOINSTRUCTION and NOASCII modes must be set. In addition, do not enclose the data within matching quotation marks or apostrophes.

You can, however, enter numeric data concurrently with instructions provided you are using the assembler directives listed below:

```
.BYTE
.WORD
.LONG
```

The following example illustrates such an insertion:

```
PATCH> DEPOSIT/PATCH_AREA/INSTRUCTION
NEW> 'TSTL R4'
NEW> 'BEQL LBL3'
NEW> 'MOVL LBL2,R0'
NEW> 'LBL1: RET'
NEW> 'LBL2: .LONG 1,0'
NEW> 'LBL3: MOVL LBL2+4,R0'
NEW> 'BRB LBL1'
```

Precede any hexadecimal number starting with an alphabetic character (that is, A through F) with a 0. For example, enter the numeric string FFA as follows:

```
0FFA
```

7.6 DELIMITING PARAMETER VALUES

The equal sign (=) is used to separate an address expression from a data entry or to separate a symbol name from a value. The commands that require the equal sign are DEFINE, DELETE, DEPOSIT, INSERT, REPLACE, and VERIFY.

To separate parameter values in commands that can optionally take multiple parameters, use a comma. For example:

```
PATCH> DEPOSIT/ASCII 508 = 'DOG','CAT','BIRD'
```

7.7 ENTERING COMMENTS

To insert comments, prefix them with the exclamation point (!) character. When PATCH reads an exclamation point, it ignores all following information on that line. The only restriction on entering comments is that a comment cannot appear on the same line as a command string that uses a continuation character (-).

The following are examples of some valid comments:

```
1.  PATCH> REPLACE/INSTRUCTION 300   !Replacing instruction in loc 300
    OLD> "PUSHAL L^11102457"         !with the new
    OLD> EXIT                        !value
    NEW> "PUSHAL L^11100245"         !PUSHAL L^1110245

2.  PATCH> DEPOSIT/ASCII-
    _> 1206 -
    _> = "CAR" -
    _> EXIT                          !This comment describes the
                                     !entire DEPOSIT command
```

7.8 TRUNCATING KEYWORDS

To simplify the process of entering commands, you can abbreviate the following keywords:

- Command names
- Mode, alignment, and other qualifiers
- Parameter values for the SET MODE command

PATCH does not recognize the different categories of keywords. Therefore, any keyword abbreviation must be unique with respect to all other keywords.

Some keywords can be truncated to nonunique abbreviations. These keyword abbreviations are listed below in Table 7-1:

Table 7-1
Nonunique Keyword Abbreviations

Keyword	Type	Abbreviation
DEPOSIT	Command name	D
EXAMINE	Command name	E
INSTRUCTION	Mode qualifier	/I
	Parameter value	I

Appendix A lists all PATCH commands with their associated qualifiers and their abbreviated format.

7.9 SPECIAL OPERATORS FOR ARITHMETIC EXPRESSIONS

This section describes the special operators that PATCH recognizes in arithmetic expressions, and gives examples of how to use these operators. Table 7-2 provides a brief description of each operator.

Table 7-2
Special Operators for Arithmetic Expressions

Operator	Name	Function
+	Addition operator	Unary plus sign or addition operator in arithmetic expressions
-	Subtraction operator	Unary minus sign or subtraction operator in arithmetic expressions
*	Multiplication operator	Multiplication operator in arithmetic expressions
/	Division operator	Division operator in arithmetic expressions
@	Shift operator	Arithmetic shift operator
< >	Priority operators	1. Priority operators 2. Bit field delimiters for the EVALUATE command
^	Radix operator	Radix operator for hexadecimal, decimal, octal, or binary notation

7.9.1 Addition Operator (+)

Use the addition operator (+) as a binary operator to add two arguments together. Use the addition operator as a unary operator to specify the existing value of the argument. Used in this context, the addition operator can be omitted and the same results will occur.

The following examples show how to use the addition operator as a binary and unary operator:

```
PATCH> EVALUATE /HEXADECIMAL ^D10 + ^D43
00000035
PATCH> DEFINE X = -1
PATCH> EVALUATE +X
0FFFFFFFF
PATCH> EVALUATE X
0FFFFFFFF
```

The results are displayed in the current radix.

7.9.2 Subtraction Operator (-)

Use the subtraction operator (-) as a binary operator to subtract one argument from another. Use the subtraction operator as a unary operator to specify the negation of the argument.

The following examples show how to use the subtraction operator as a binary and unary operator:

```
PATCH> EVALUATE /HEXADECIMAL ^D58 - ^D14
0000002C
PATCH> DEFINE X = -1
PATCH> EVALUATE -X
00000001
```

The results are displayed in the current radix.

7.9.3 Multiplication Operator (*)

Use the multiplication operator (*) to multiply two arguments. For example:

```
PATCH> EVALUATE 10 * 10
00000100
```

The result is displayed in the current radix.

7.9.4 Division Operator (/)

Use the division operator (/) to divide the first argument by the second argument. PATCH truncates the quotient to an integer value. For example:

```
PATCH> EVALUATE ^D5 / ^D3
00000001
```

The result is displayed in the current radix.

Note that PATCH does not let you divide by 0.

7.9.5 Arithmetic Shift Operator (@)

Use the arithmetic shift operator (@) to shift the first argument the number of bit positions specified by the following value. A positive value means shift to the left and lose the contents of the sign bit. A negative value means shift to the right and propagate the contents of the sign bit in the vacated bit positions.

The shift operation is arithmetic; no wraparound occurs as it does in a logical shift.

The following example demonstrates the use of the arithmetic shift character:

```
PATCH> EVALUATE 5FC0 @-2
000017F0
PATCH> EVALUATE ^D100 @2
00000190
```

The results are displayed in the current radix.

7.9.6 Priority Operator (< >)

Expressions are normally evaluated from left to right, except when the command contains angle brackets, or priority operators, (< >). Then, the expression contained within the priority operators is evaluated first. An expression can contain up to 16 levels of nested expressions. PATCH evaluates them from innermost to outermost. For example:

```
PATCH> EVALUATE/DECIMAL <5*<716-<5+89>>>
3110
```

7.9.7 Radix Operator (^)

Use the radix operator to control PATCH's interpretation of numeric strings. The radix character affects only the entry that it accompanies; PATCH displays the results in the current radix.

The radix operators for the VAX-11 MACRO language are:

Operator	Radix
^O	Octal radix
^D	Decimal radix
^X	Hexadecimal radix
^B	Binary radix

PATCH does not allow spaces between a radix character and a numeric string.

The following example uses a radix character:

```
PATCH> DEPOSIT/DECIMAL ^X300 = 15
old:     00000768:     00000000
new:     00000768:     00000015
```

In the above example, the radix mode is set to decimal representation; however, the location to be modified is set to hexadecimal representation. The display is decimal.

7.10 SPECIAL OPERATORS FOR ADDRESSING LOCATIONS

This section describes the special operators that PATCH recognizes for locating addresses and gives examples of how to use these operators. Table 7-3 provides a brief description of each operator.

Table 7-3
Special Operators for Addressing

Operator	Name	Function
.	Current location operator	Displays the location last specified; that is, the current location
^	Previous location operator	Displays the location that precedes the last location specified; (subtract, from the current location, the number of bytes indicated by the length mode)
:	Range operator	Range operator for the CHECK ECO, CHECK NOT ECO, EVALUATE, and EXAMINE commands
\	Backslash operator	Displays either the contents of the location specified in a branch instruction or the value stored in an address specified in the previous EXAMINE command

7.10.1 Current Location Operator (.)

Use the current location operator (.) to access the address specified most recently. This value remains the same until you reference a new address. The current location operator can be used in place of a symbolic or numeric address in any command that requests a location. For example:

```
PATCH> EXAMINE/INSTRUCTION 500
00000500:     BBC RMS_DOC
PATCH> DEPOSIT/INSTRUCTION .  = 'BBCC RMS_DOC'
```

7.10.2 Previous Location Operator (^)

Use the previous location operator (^) to access the address preceding the current address. The previous location operator can be used in place of a symbolic or numeric address in any command that requests a location. Using this operator when INSTRUCTION mode is set produces unpredictable results.

The following example demonstrates the use of the previous location operator:

```
PATCH> EXAMINE/ASCII/LONG 506
00000506:     'RATH'
PATCH> EXAMINE/ASCII/LONG ^
00000502:     'VERA'
```

Note that PATCH decrements memory locations according to the current length mode.

7.10.3 Range Operator (:)

Use the range operator (:) to specify a range of locations for the EXAMINE command or to specify a range of ECO levels in the CHECK ECO and CHECK NOT ECO commands. For example:

```
PATCH> EXAMINE .:205
00000203:     TSTL R4
00000205:     BNEQ PAD$STRT
PATCH> CHECK ECO 99:103
```

The range operator can also be used to separate bit field delimiters in the EVALUATE command. For example:

```
PATCH> EVALUATE 700<8:5>
```

See the EVALUATE command in Chapter 8 for a complete description of evaluating variable length bit fields.

7.10.4 Backslash Operator (\)

Use the backslash operator (\) with the EXAMINE/INSTRUCTION command to display the contents of a location specified in a branch instruction. For example:

```
PATCH> EXAMINE/INSTRUCTION 750
00000750:  BRW 7515
PATCH> EXAMINE/INSTRUCTION \
00007515:  PUSHL L^NAME\BEGIN+12,L^NAME\BEGIN+23
```

Used with the EXAMINE command, the backslash operator displays the value stored in an address that is displayed in response to the previous EXAMINE command. For example:

```
PATCH> EXAMINE 100
00000100:    00000500
PATCH> EXAMINE \
00000500:    00000002
```

7.11 VAX-11 MACRO INSTRUCTIONS WITH THE SAME OPCODES

IN VAX-11 MACRO, different instructions can generate the same opcode. For example, MOVAL and MOVAF both generate the hexadecimal opcode DE. PATCH, however, displays only one instruction for any of the opcodes that can be produced by multiple instructions. For example, if you use MOVAF in your source program, PATCH displays MOVAL.

Table 7-4 lists the instructions that have the same opcodes, lists the instruction produced by PATCH, and lists the opcodes themselves.

Table 7-4
Instructions Displayed by PATCH for
VAX-11 MACRO Instructions with Equivalent Opcodes

Instructions with Equivalent Opcodes	Instruction Displayed by PATCH	Opcode
BCC BGEQU	BGEQU	1E
BCS BLSSU	BLSSU	1F
BEQL BEQLU	BEQL	13
BNEQ BNEQU	BNEQ	12
CLRD CLRG CLRQ	CLRQ	7C
CLRF CLRL	CLRL	D4
CLRH CLRO	CLRO	7CFD
MOVAD MOVAG MOVAQ	MOVAQ	7E
MOVAF MOVAL	MOVAL	DE
MOVAH MOVAO	MOVAO	7EFD
PUSHAD PUSHAG PUSHAQ	PUSHAQ	7F
PUSHAF PUSHAL	PUSHAL	DF
PUSHAH PUSHAO	PUSHAO	7FFD

CHAPTER 8

PATCH COMMANDS

This chapter lists and describes the PATCH commands in alphabetical order. For a summary description of the PATCH commands, see Appendix A.

ALIGN

Description

Use ALIGN to allocate space for default patch area in image files that have not been patched previously or to advance the starting address of the current patch area to the first free byte aligned on the requested boundary (byte, word, longword, quadword, or page) and to equate a symbolic name to that address. Once you define the symbolic name, you can use it in place of the address it denotes.

When you specify ALIGN, none of the patch area between the old patch area address and the aligned patch area address is cataloged by PATCH. You must keep a record of the disjointed patch area for it to be used.

After an alignment has been made, the patch area string descriptor is updated to reflect the modifications. If the patch area is already aligned on the specified boundary, no update is performed. (See Section 5.3 for more information on patch area string descriptors.)

You must enter one, and only one, alignment qualifier for each ALIGN command.

Format

ALIGN symbol-name

Alignment Qualifiers	Defaults
/BYTE	None.
/WORD	
/LONG	
/QUAD	
/PAGE	

Prompts

NAM> symbol-name

Parameters

symbol-name

Specifies a 1- to 15-character symbol. It must start with an alphabetic character, and consist of alphanumeric characters, dollar signs ($), and/or underscores (_).

If you specify a symbol name for an address that has been previously assigned a symbol name, the newest symbol name takes precedence.

Alignment Qualifiers

/BYTE

Defines the symbol as the first free byte of the current patch area. If the current patch area has not been used previously, PATCH allocates the first block of default patch area.

/WORD

Advances the starting address of the current patch area to the first free word boundary.

/LONG

Advances the starting address of the current patch area to the first free longword boundary.

/QUAD

Advances the starting address of the current patch area to the first free quadword boundary.

/PAGE

Advances the starting address of the current patch area to the first free page boundary.

Examples

1.
```
PATCH> ALIGN/QUAD MOD_1
old patch area size:          000001E3
old patch area address:       0000081D
new patch area size:          000001E0
new patch area address:       00000820
symbol "MOD_1" defined as     00000820
```

The ALIGN command requests that the patch area starting address be advanced to the first free quadword boundary in the current patch area and that the symbol MOD_1 be equated with that address. The display shows the old and new patch area status.

2.
```
PATCH> ALIGN/BYTE PATAREA
old patch area size:          00000000
old patch area address:       00000000
new patch area size:          00000200
new patch area address:       00001800
symbol "PATAREA" defined as   00001800
```

This ALIGN command allocates the first block for the default patch area and assigns the symbol PATAREA to its new starting address.

CANCEL MODE

Description

Use the CANCEL MODE to control the syntax of commands you enter and the values PATCH displays. CANCEL MODE cancels the current mode settings and reinstates the initial default mode settings. The initial default mode settings are: NOINSTRUCTION, NOASCII, SYMBOLS, HEXADECIMAL, LONG, NOGLOBALS, and SCOPE.

Format

CANCEL MODE

Prompts

None.

Parameters

None.

Examples

1.
```
PATCH> SHOW MODE
modes:  nosymbols, instruction, ascii, scope, globals, decimal, word
PATCH> CANCEL MODE
PATCH> SHOW MODE
modes:  symbols, noinstruction, noascii, scope, noglobals, hexadecimal long
```

The first SHOW MODE command displays the current mode status. The CANCEL MODE command requests that the initial default mode settings be reinstated. The second SHOW MODE command confirms that the initial default settings have been re-established.

CANCEL MODULE

Description

Use CANCEL MODULE to remove local symbol information from PATCH's symbol table; this command does not remove global symbols, patch area symbols, universal symbols, or symbols you defined with the DEFINE command. Once the module's local symbols have been removed, they cannot be used to reference locations.

To remove all local symbol information from the symbol table, specify the /ALL qualifier.

If the scope is the name of the module that you specify with the CANCEL MODULE command, the contents of the scope is reset to the empty string (<null>).

Format

CANCEL MODULE module-name [,...]

Other Qualifiers	Defaults
/ALL	None.

Prompts

NAM> module-name

Parameters

module-name

Specifies the name of one or more modules whose local symbols are to be removed from the symbol table.

Do not specify a module name if you include the /ALL qualifier.

Other Qualifiers

/ALL

Removes all local symbol information from the symbol table.

Examples

1.
```
PATCH> SHOW MODULE
module name          symbols      size

LOVAT                yes          180.

total modules:       1.
remaining size:      EEA4.
PATCH> CANCEL MODULE
NAM> LOVAT
NAM> EXIT
```

The SHOW MODULE command indicates that the local symbols contained in the module named LOVAT are entered in the symbol table. The CANCEL MODULE command purges the symbol table of the local symbols contained in LOVAT.

2. PATCH> CANCEL MODULE/ALL

This CANCEL MODULE command removes all local symbol names entered in the symbol table.

3. PATCH>SET SCOPE CIRCLE

```
      .
      .
      .
PATCH> CANCEL MODULE CIRCLE
PATCH> SHOW SCOPE
SCOPE:  <null>
```

The SET SCOPE command establishes the module named CIRCLE as the current scope setting. Later, the CANCEL MODULE command removes all local symbols in CIRCLE from the symbol table and, at the same time, sets the scope to the empty string (<null>). The SHOW SCOPE command confirms the new scope setting.

CANCEL PATCH_AREA

Description

Use CANCEL PATCH_AREA to reset the current patch area from a user-defined patch area to the default patch area. Any patch area needed thereafter will be taken from the default patch area until the next SET PATCH_AREA command is issued.

The CANCEL PATCH_AREA command is used primarily when you have issued the SET PATCH_AREA command to establish a user-defined patch area as the current patch area. After you have inserted the necessary patch information into that area, type CANCEL PATCH_AREA to resume use of the default patch area.

You must specify the CANCEL PATCH_AREA command if you have defined, and are using, a user-defined patch area that is too small to store the patches you want to insert. Failing to use the CANCEL PATCH_AREA command causes PATCH to return an error message indicating that it cannot insert the patch, from the failing command, into the patch area.

Format

CANCEL PATCH_AREA

Prompts

None.

Parameters

None.

Examples

1.
```
PATCH> SET PATCH_AREA AREA1
     .
     .
     .
PATCH>CANCEL PATCH_AREA
```

The SET PATCH_AREA command establishes the user-defined patch area AREA1 as the current patch area. After depositing the necessary data into AREA1, the CANCEL PATCH_AREA command is issued to resume use of the default patch area.

CANCEL SCOPE

Description

Use CANCEL SCOPE to cancel the current symbolic scope and revert to an empty string (<null>).

Format

CANCEL SCOPE

Prompts

None.

Parameters

None.

Examples

1.
```
PATCH> SHOW SCOPE
SCOPE: MOD1
PATCH> CANCEL SCOPE
PATCH> SHOW SCOPE
scope: <null>
```

The first SHOW SCOPE command indicates that the scope is set to the module named MOD1. The CANCEL SCOPE command cancels the scope setting and reverts to the empty string (<null>). The second SHOW SCOPE command confirms that the contents of the scope is an empty string.

CHECK ECO

Description

Use CHECK ECO to check that one or more ECO levels have been set, and therefore, that the patches represented by these ECO levels have been applied to the image file. Remember that an ECO level is not set until the patch that it represents is terminated by the UPDATE command. For example, if you define an ECO level with the SET ECO command, then immediately check to see whether the ECO level is set, PATCH returns an error message indicating that the ECO level is not set.

When you issue the CHECK ECO command, you do not have to list all the ECO levels that have been set for a particular image file. However, specifying one or more ECO levels that have not been set will produce an error message.

Format

CHECK ECO eco-level [:eco-level] [,...]

Prompts

ECO> eco-level

Parameters

eco-level

Indicates one or more ECO levels that have been set. ECO levels can be entered in:

- Comma-separated lists
- Colon-separated ranges

Both comma-separated lists and colon-separated ranges can be specified on the initial command line. However, only one ECO level or one range of ECO levels can be entered in response to an ECO level prompt (ECO>).

Examples

1.
```
PATCH> CHECK ECO   3:8,14,16
```

This command checks that the patches associated with ECO levels 3, 4, 5, 6, 7, 8, 14, and 16 have been applied to the image file.

2.
```
PATCH> CHECK ECO 12
%PATCH-E-ECONOTSET, eco level 12 not set in DB2:[HARINGTON]BIND_NOW.EXE;4
```

The CHECK ECO command checks that the patch associated with ECO level 12 has been applied to the image file. The display indicates that ECO level 12 has not been set and that the patch it represents has not been applied to the image file BIND.EXE.

CHECK NOT ECO

Description

Use CHECK NOT ECO to check that one or more ECO levels are available for use in a particular image file.

The CHECK NOT ECO command is the negation of the CHECK ECO command. It too can be used to confirm that a particular patch has been applied to an image file. Typically, however, the CHECK NOT ECO command is used to confirm the ECO levels that are available for use in the current image file.

Format

CHECK NOT ECO eco-level [:eco-level] [,...]

Prompts

ECO> eco-level

Parameters

eco-level

Indicates the ECO levels that are not set. ECO levels can be entered in:

- Comma-separated lists
- Colon-separated ranges

Both comma-separated lists and colon-separated ranges can be entered on the initial command. However, only one ECO level or one range of ECO levels can be entered in response to an ECO level prompt (ECO>).

Examples

1.
```
PATCH> CHECK NOT ECO   4:6,10
```

This command confirms that the ECO levels 4, 5, 6, and 10 are available for use.

2.
```
PATCH> CHECK NOT ECO
ECO> 17
ECO> EXIT
%PATCH-E-ECOSET, eco level 17 already set in DB1:[REAVER]MYFILE.EXE;7
```

In response to the CHECK NOT ECO command, PATCH indicates that ECO level 17 has already been used in the image file MYFILE.EXE.

CREATE

Description

Use CREATE to create a command procedure that contains all PATCH commands successfully executed after the CREATE command.

Command procedures facilitate patching multiple copies of the same image file. There are two ways to create command procedures. One way is to specify the CREATE command when you invoke PATCH. All subsequent, successful commands are applied to the image file and recorded in the command procedure. A second way is to use a text editor. This method is discussed in Section 3.4.2.

When you use CREATE to create a command procedure, PATCH automatically inserts, as the first entry in the command procedure, the name of the image file that will incorporate the patches. All symbolic names are converted to absolute values, and all command names and qualifiers are truncated to their shorthand notation.

You can issue only one CREATE command per PATCH session. To create another command procedure, close the input image file and then reopen it.

To process the patches contained in the command procedure, issue the following DCL command:

```
$ PATCH @file-spec
```

In the above command line, the file-spec represents the specification of the command procedure containing the patches.

Chapter 2 discusses the different techniques for creating command procedures; Chapter 3 discusses how to submit command procedures for PATCH processing.

Format

```
CREATE   [file-spec]
```

Prompts

None.

Parameters

file-spec

Represents the specification of the command procedure. The syntax is:

```
device:[directory]filename.filetype;version
```

You can omit all or some of the fields in the command procedure file specification. PATCH uses the default values listed below for omitted fields.

Field	Default Value
device:[directory]	The process's current default device and directory
filename	The name of input image file
filetype	.COM
version	1 greater than the highest command procedure of the same name

Note that if you store a command procedure in a directory other than the one that contains the input image file to which the command procedure is applied, you must set your default to the directory that contains the input image file before you process the command procedure.

Examples

1.
```
$ PATCH AVERAGE

  PATCH VERSION 2.0 15 MAR 80

PATCH> CREATE PAT2
PATCH> SET ECO 1
PATCH> SET MODE NOSYMBOLS
PATCH> EXAMINE 600
00000600:     12345678
PATCH> DELETE 600 = 12345678
old:     00000600:     12345678
new:     00000600:     00000000
PATCH> UPDATE
%PATCH-I-WRTFIL, updating image file DBA2:[BRADLEY]AVERAGE.EXE;6
PATCH> EXIT
$ SET DEFAULT [NIMROD]
$ SHOW DEFAULT
DBA2:[NIMROD]
$ PATCH @[BRADLEY]PAT2
```

In the above command procedure, PATCH is invoked to patch the image AVERAGE.EXE in the DBA2:[BRADLEY] directory. The CREATE command creates the command procedure PAT2.COM in which all successful commands are stored.

After the PATCH session ends, the default directory is set to DBA2:[NIMROD] and the patches in the command procedure PAT2.COM are applied to the image file [NIMROD]AVERAGE.EXE.

DEFINE

Description

Use DEFINE to equate a symbolic name to a specific value and place the symbolic name in the symbol table. Once the assignment has been performed, you can specify the the symbolic name in place of the value it denotes for the duration of the PATCH session. At the end of the PATCH session, these user-defined symbols are deleted from the symbol table.

When you use the DEFINE command to create symbolic names, PATCH always searches the symbol table for these symbolic names first when it translates a symbol into a value.

More than one symbolic name can be assigned to a single value. Each symbolic name is recorded in the symbol table and can subsequently be used to reference the value it denotes.

You can redefine a symbolic name to represent a new value. Then, when you specify the symbolic name, the most recent value that the symbol denotes is displayed.

Two restrictions apply to the use of the DEFINE command:

- You can not equate a symbolic name to a pathname
- You cannot specify the /INSTRUCTION or /ASCII mode qualifiers, nor can you set the INSTRUCTION or ASCII modes, when equating a symbol name to a value

Format

```
DEFINE symbol-name = value [,symbol-name = value,...]
```

Prompts

```
NAM> symbol-name
VAL> value
```

Note that if you enter the symbol name on the initial command line, you must also enter the value on that line.

Parameters

symbol-name

Specifies a 1- to 15-character user-defined symbol to be associated with the specified value. The symbol name must start with an alphabetic character, and can consist of alphanumeric characters, dollar signs ($), and/or underscores (_). (PATCH does not distinguish between uppercase and lowercase letters; that is, the value ABC is equivalent to the value abc.)

The symbol name cannot be a pathname.

value

Specifies a numeric address or symbolic expression that is to be assigned the specified symbolic name.

Examples

1.
```
PATCH> DEFINE SWEDISH = CAR\VOLVO+144
Symbol "SWEDISH" defined as CAR\VOLVO+144
PATCH> EXAMINE SWEDISH
```

The DEFINE command creates a symbolic name for the value CAR\VOLVO+144. A subsequent EXAMINE command requests to see the contents of CAR\VOLVO+144 using the symbol name SWEDISH.

2.
```
PATCH> DEFINE
NAM> SECOND_CHOICE
VAL> 406
NAM> EXIT
symbol "SECOND_CHOICE" redefined from 408 to 406
```

The DEFINE command reassigns the symbol SECOND_CHOICE from the value 408 to the value 406. The DEFINE command response shows the reassignment.

DELETE

Description

Use DELETE to delete an instruction or piece of data from one location or from several consecutive locations in terms of the current mode settings.

When you use the DELETE command to delete instructions, the instructions are replaced with NOP instructions. When you use the DELETE command to delete ASCII and numeric data, the data is replaced with zeros.

Format

```
DELETE location = current-contents [,...]
```

Mode Qualifiers	Defaults
/BYTE /WORD /LONG	/LONG
/OCTAL /DECIMAL /HEXADECIMAL	/HEXADECIMAL
/[NO]ASCII /[NO]INSTRUCTION /[NO]SYMBOLS	/NOASCII /NOINSTRUCTION /SYMBOLS
/[NO]GLOBALS /[NO]SCOPE	/NOGLOBALS /SCOPE

Prompts

```
LOC> location
OLD> current-contents
```

Parameters

location

Specifies either (1) a single location whose contents are to be deleted or (2) the starting address of a sequence of locations whose contents are to be deleted.

current-contents

Specifies one or more data entries or instructions to be deleted. The data or instructions you specify must be the actual contents.

Do not specify conflicting data types within a single DELETE command.

Mode Qualifiers

/BYTE
/WORD
/LONG

Determine whether data is deleted in byte, word, or longword lengths. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the DELETE command ignores the current length setting and deletes the entire instruction.

The initial default setting is /LONG.

/OCTAL
/DECIMAL
/HEXADECIMAL

Determine the radix in which virtual addresses and numeric data are interpreted and displayed. The radix mode qualifiers do not affect symbolic addresses and instructions, or ASCII data.

The initial default setting is /HEXADECIMAL.

/ASCII
/NOASCII

Control whether the data to be deleted is interpreted and displayed as ASCII data.

When you specify /ASCII, enclose the data within matching quotation marks or apostrophes. The current radix setting has no affect on the ASCII data being deleted; however, the DELETE command truncates the data if it exceeds the length imposed on it by the current length mode.

The initial default setting is /NOASCII.

/INSTRUCTION
/NOINSTRUCTION

Control whether the data to be deleted is interpreted and displayed as VAX-11 MACRO instructions.

When you specify /INSTRUCTION, enclose the instruction within matching quotation marks or apostrophes. Neither the current radix setting nor the current length setting affect the instruction being deleted.

The initial default setting is /NOINSTRUCTION

/SYMBOLS
/NOSYMBOLS

Control whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/GLOBALS
/NOGLOBALS

Control whether the symbolic entry (exactly as entered) is used as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/SCOPE
/NOSCOPE

Control whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

Examples

1.
```
PATCH> DELETE/INSTRUCTION   2112 = 'CMPB (R0),(R5)'
old: 00002112: CMPB (R0),(R5)
new: 00002112:   NOP
new: 00002113:   NOP
new: 00002114:   NOP
```

The DELETE command replaces the instruction CMPB (R0),(R5) with NOP instructions.

2.
```
PATCH> DELETE/SYMBOLS
LOC> 7A6
OLD> 0E6ADDE39
OLD> EXIT
old: OTS$LINKAGE\OTS$LINKAGE+7C    0E6ADDE39
new: OTS$LINKAGE\OTS$LINKAGE+7C    00000000
```

The DELETE command deletes the contents of location 7A6. Because /SYMBOLS is specified, location 7A6 is reported symbolically.

DEPOSIT

Description

Use DEPOSIT to deposit new data or instructions into one or more consecutive locations.

Depositing Data or Instructions: The DEPOSIT command lets you replace the contents of a location or of several consecutive locations in terms of the current mode settings. The DEPOSIT command does not request verification of the current contents before replacing the contents with new data or instructions (that is, this command assumes that you know what you are doing). In cases when you want to confirm that data or instructions that will be overwritten, use the REPLACE command.

When you are adding instructions to an image file, it is easier to use the INSERT command. This command performs automatic branching to and from the patch area.

Depositing Data or Instructions into Patch Area: When you append the /PATCH_AREA qualifier to the DEPOSIT command, the data or instructions specified are inserted into the current patch area. The location you supply must be the first free byte in patch area. To determine the first free byte in patch area, issue the SHOW PATCH_AREA command or the ALIGN/BYTE command. After you deposit the data, PATCH updates the patch area string descriptor to reflect the modifications.

Unlike the INSERT and REPLACE commands, the DEPOSIT/PATCH_AREA command requires that you insert the branch instructions into the appropriate locations to maintain the logical flow of program execution to and from the patch area.

See Chapter 5 for more information on using patch area.

Format

DEPOSIT location = new-contents [,...]

Mode Qualifiers	Defaults
/BYTE /WORD /LONG	/LONG
/OCTAL /DECIMAL /HEXADECIMAL	/HEXADECIMAL
/[NO]ASCII	/NOASCII
/[NO]INSTRUCTION	/NOINSTRUCTION
/[NO]SYMBOLS	/SYMBOLS
/[NO]GLOBALS	/NOGLOBALS
/[NO]SCOPE	/SCOPE

Other Qualifiers

/PATCH_AREA

Prompts

LOC> location
NEW> new-contents

Parameters

location

Specifies either (1) a single location whose contents are to be overwritten or (2) the starting address of a sequence of locations whose contents are to be overwritten.

new-contents

Specifies one or more data entries or instructions to be inserted. Do not enter conflicting data types with a single DEPOSIT command.

Mode Qualifiers

/BYTE
/WORD
/LONG

Determine whether data is deposited in byte, word, or longword lengths. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the DEPOSIT command ignores the current length mode and deposits the entire instruction.

The initial default setting is /LONG.

/OCTAL
/DECIMAL
/HEXADECIMAL

Determine the radix in which virtual addresses and numeric data are interpreted and displayed. The radix mode qualifiers do not affect symbolic addresses and instructions, or ASCII data. The initial default setting is /HEXADECIMAL.

/ASCII
/NOASCII

Control whether the data to be deposited is interpreted and displayed as ASCII data.

When you specify /ASCII, enclose the data within matching quotation marks or apostrophes. The current radix setting has no affect on the ASCII data being deposited; however, the DEPOSIT command truncates the data if it exceeds the length imposed on it by the current length mode.

The initial default setting is /NOASCII.

/INSTRUCTION
/NOINSTRUCTION

Control whether the data to be deposited is interpreted and displayed as VAX-11 MACRO instructions

When you specify /INSTRUCTION, enclose the instruction within matching quotation marks or apostrophes. Neither the current radix setting nor the current length setting affects the instruction being deposited.

The initial default setting is /NOINSTRUCTION.

/SYMBOLS
/NOSYMBOLS

Control whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/GLOBALS
/NOGLOBALS

Control whether the symbolic entry (exactly as entered) is used as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/SCOPE
/NOSCOPE

Control whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

Other Qualifiers

/PATCH_AREA

Signals PATCH to deposit the data or instructions into the current patch area, starting at the specified location.

Examples

1.
```
PATCH> DEPOSIT/ASCII/BYTE 1111 = 'A'
old: 00001111:    'B'
new: 00001111:    'A'
```

The DEPOSIT command requests that a byte of ASCII data be deposited in location 1111. The DEPOSIT display indicates that the data was successfully deposited and that the old contents was B.

2.
```
PATCH> DEPOSIT/INSTRUCTION
LOC> 413
NEW> 'CMPW (R1), R6'
NEW> EXIT
old: 00000413:    CMPW (R6), R1
new: 00000413:    CMPW (R1), R6
```

The DEPOSIT command deposits an instruction into location 413. The /INSTRUCTION qualifier is specified to indicate that an instruction is being deposited. The DEPOSIT display indicates that the instruction was successfully deposited.

3.
```
PATCH> SET PATCH_AREA NEW_PATCH
PATCH> ALIGN/BYTE
old patch area size:          0000018C
old patch area address:       000004A8
new patch area size:          0000018C
new patch area address:       000004A8
symbol "PAT1" defined as:     000004A8
PATCH> DEPOSIT/PATCH_AREA/ASCII
LOC> PAT1
NEW> 'RUN'
NEW> 'SKIP'
NEW> EXIT
old:  PAT1:   "
old:  000004AC:
new:  PAT1:    'RUN'
new:  000004AC:    'SKIP
```

The SET PATCH_AREA command establishes the user-defined patch area named NEW_PATCH as the current patch area. The ALIGN/BYTE command realigns the starting address of NEW_PATCH on the first available byte address and defines the symbol PAT1 to that address. The DEPOSIT/PATCH_AREA command is then issued to deposit into NEW_PATCH at PAT1 the ASCII data RUN and SKIP.

EVALUATE

Description

Use EVALUATE to evaluate one of the following:

- Arithmetic expressions
- Values
- Variable-length bit fields

Evaluating Arithmetic Expressions: Use the EVALUATE command to perform binary and unary arithmetic operations. (These operations are discussed in Section 7.8.)

The EVALUATE command interprets expressions and displays results in the current length and radix modes.

Evaluating Values: Use the EVALUATE command to determine the value associated with a symbol or pathname. The values are displayed in terms of the current length and radix mode setting.

Evaluating Variable-Length Bit Fields: Use the EVALUATE command to display the current contents of a specific bit field in a value. The syntax for this command is:

```
PATCH> EVALUATE value <high-bit:low-bit>
```

The bit position delimiters (high-bit and low-bit) are specified as decimal integers.

Bit positions range from 0 (least significant bit) to 31 (most significant bit). PATCH extracts the contents of the bit positions and reports the contents in longword representation and in terms of the current radix setting. The current length mode is ignored. Note that ASCII mode and INSTRUCTION mode cannot be set when you evaluate selected bit positions.

Format

EVALUATE expression [,...]

Mode Qualifiers	Defaults
/BYTE /WORD /LONG	/LONG
/OCTAL /DECIMAL /HEXADECIMAL	/HEXADECIMAL
/[NO]ASCII /[NO]INSTRUCTION	/NOASCII /NOINSTRUCTION
/[NO]GLOBALS /[NO]SCOPE	/NOGLOBALS /SCOPE

Prompts

EXP> expression

Parameters

expression

Indicates an arithmetic expression, a value and corresponding bit field, or a literal value that is to be evaluated in terms of the current mode settings. As explained above, when you evaluate a selected bit field in a value, the format of the expression is:

value <high-bit:low-bit>

Mode Qualifiers

/BYTE
/WORD
/LONG

Determine whether data is evaluated in byte, word, or longword lengths. The length mode qualifiers affect only the evaluation of arithmetic expressions; they have no control over evaluating bit fields.

The initial default setting is /LONG.

/OCTAL
/DECIMAL
/HEXADECIMAL

Determine the radix in which data is interpreted and displayed. The radix mode qualifiers affect the evaluation of arithmetic expressions and of bit fields for specific values.

The initial default setting is /HEXADECIMAL.

/ASCII
/NOASCII

Control whether the data is interpreted as ASCII data. The /ASCII mode qualifier or the ASCII mode cannot be set when evaluating variable length bit fields.

The initial default setting is /NOASCII.

/INSTRUCTION
/NOINSTRUCTION

Control whether data is interpreted as VAX-11 MACRO instructions. The /INSTRUCTION mode qualifier or the INSTRUCTION mode cannot be set when evaluating variable-length bit fields.

The initial default setting is /NOINSTRUCTION.

/GLOBALS
/NOGLOBALS

Control whether the symbolic entry (exactly as entered) is used as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/SCOPE
/NOSCOPE

Control whether scope's contribution to a pathname is used to find the location specified.

The intitial default setting is /SCOPE.

Examples

1.
```
PATCH> EVALUATE/DECIMAL 101 * 123
12423
```

This command performs the requested arithmetic operation. The values are interpreted and displayed in decimal representation.

2.
```
PATCH> SET MODE DECIMAL
PATCH> EVALUATE ^X200 + < ^D65 / ^012 >
518
```

The EVALUATE command performs the specified arithmetic operation according to the rules of precedence. PATCH first divides decimal 65 by octal 12, and then adds the quotient to hexadecimal 200. The result is displayed in decimal representation.

3.
```
PATCH> EVALUATE ^070 <5:3>
00000007
```

This command calculates the specified bit field value for the octal number 70. The result is displayed in the current radix setting (in this case, hexadecimal).

4.
```
PATCH> EVALUATE FDR$PETE
00000800
```

The EVALUATE command determines that the value assigned to the symbol FDR$PETE is 800. The result is displayed in the current radix setting (in this case, decimal).

EXAMINE

Description

Use EXAMINE to display the contents of the specified locations in terms of the current mode settings.

You can also use the EXAMINE command to examine the contents of a branch instruction or the contents of an address displayed in response to the previous EXAMINE command. To do so, you use the backslash operator (\), which is fully described in Section 7.10.4.

Format

EXAMINE location [:location] [,...]

Mode Qualifiers	Defaults
/BYTE /WORD /LONG	/LONG
/OCTAL /DECIMAL /HEXADECIMAL	/HEXADECIMAL
/[NO]ASCII /[NO]INSTRUCTION /[NO]SYMBOLS	/NOASCII /NOINSTRUCTION /SYMBOLS
/[NO]GLOBALS /[NO]SCOPE	/NOGLOBALS /SCOPE

Prompts

None.

Parameters

location

Specifies one or more locations whose contents are to be displayed. Multiple locations can be specified in a comma-separated list or colon-separated range. Both comma-separated lists and colon-separated ranges can be specified on a single command line.

The location parameter can also be represented by the backslash operator (\). This operator is used to observe the contents of the location displayed in the previous EXAMINE command.

If you do not supply a location, the contents of the next sequential location will be displayed.

Mode Qualifiers

/BYTE
/WORD
/LONG

Determine whether data is displayed in byte, word, or longword lengths. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the EXAMINE command ignores the current length setting and displays the entires instruction.

The initial default setting is /LONG.

/OCTAL
/DECIMAL
/HEXADECIMAL

Determine the radix in which virtual addresses are interpreted and numeric data is displayed. The radix mode qualifiers do not affect symbolic addresses, symbolic instructions, or ASCII data.

The initial default setting is /HEXADECIMAL.

/ASCII
/NOASCII

Control whether data is displayed as ASCII data.

When you specify /ASCII, the EXAMINE command truncates the data if it exceeds the limit imposed on it by the current length setting.

The initial default setting is /NOASCII.

/INSTRUCTION
/NOINSTRUCTION

Control whether data is displayed as VAX-11 MACRO instructions.

When you specify /INSTRUCTION, the EXAMINE command ignores both the current radix setting and the current length setting when displaying an instruction.

The initial default setting is /NOINSTRUCTION.

/SYMBOLS
/NOSYMBOLS

Control whether locations are displayed as pathnames or symbols, rather than numeric addresses.

The initial default setting is /SYMBOLS.

/GLOBALS
/NOGLOBALS

Control whether the symbolic entry (exactly as entered) is used as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/SCOPE
/NOSCOPE

Control whether scope's contribution to a pathname is used to find the location specified.

The initial default setting is /SCOPE.

Examples

1.
```
PATCH> EXAMINE/INSTRUCTION/NOSYMBOLS 600
00000600:       BNEQ 634
PATCH> EXAMINE/INSTRUCTION/NOSYMBOLS \
00000634:       TSTL R4
```

The response to the first EXAMINE command indicates that memory location 600 contains a branch instruction to location 634. Because the backslash character (\) is specified in the second EXAMINE command, the contents of location 634 are displayed. The /NOSYMBOLS qualifier requests that the locations be displayed by virtual addresses.

2.
```
PATCH> EXAMINE/ASCII 650:.
00000650:       'FE  '
00000654:       'FI  '
00000658:       'FO  '
0000065C:       'FUM '
```

THE EXAMINE command requests a display in ASCII of all the data between location 650 and the current location (represented by the dot character).

3.
```
PATCH> SET MODE SCOPE,SYMBOLS
PATCH> SET SCOPE NEGATION
     .
     .
     .

PATCH> SET MODE NOSCOPE
     .
     .
     .

PATCH> EXAMINE/INSTRUCTION/SCOPE RO_CODE
NEGATION/RO_CODE:  PUSHL L^NEGATION/RW_DATA+14
```

In the above sequence of commands, the modes SCOPE and SYMBOLS are set and the scope is established as the module named NEGATION. After PATCH performs a series of commands, the SCOPE-NOSCOPE mode is turned off, although the scope is still set to NEGATION.

Still later, the EXAMINE command is specified to request the display of the instruction in location RO_CODE. The /SCOPE qualifier requests that the scope be prefixed to RO_CODE and that the symbol table be searched for a value that matches that entry.

4.
```
PATCH> EXAMINE/INSTRUCTION    .
00000600:    BNEQ  634
PATCH> EXAMINE/INSTRUCTION
00000602:    BRW   LPLIST
PATCH> EXAMINE/INSTRUCTION    \
LPLIST:  MOVL R3,R4
```

The first EXAMINE command requests that the contents of the current location (represent by the dot character) be displayed as a VAX-11 MACRO instruction. The second EXAMINE command requests that the next sequential location be displayed as a VAX-11 MACRO instruction. The third EXAMINE command requests that the contents of the destination of the previous branch instruction (BRW) be displayed.

EXIT

Description

Use EXIT to terminate a repetitive prompt such as NEW>, OLD>, or ECO>, or to terminate a PATCH session and pass control back to the command interpreter. You can also specify CTRL/Z to terminate a PATCH session. Issue CTRL/Z in response to the PATCH prompt (PATCH>).

Do not type EXIT in response to the value prompt (VAL>) for the DEFINE command. For this command, only the name prompt (NAM>) recognizes the EXIT command.

Format

EXIT

Prompts

None.

Parameters

None.

Examples

1.
```
PATCH> SET MODE
NEW> INSTRUCTION
NEW> NOSYMBOLS
NEW> NOSCOPE
NEW> EXIT
PATCH>
```

The SET MODE command continuously prompts for new mode settings until you enter the EXIT command.

2.
```
PATCH> EXIT
$
```

When you specify the EXIT command in response to a PATCH prompt (PATCH>), the PATCH session is terminated and control is passed back to the command interpreter. The dollar sign ($) indicates that you are no longer in PATCH.

3.
```
PATCH> DEFINE
NAM> TEST1
NEW> 500
NAM> TEST2
NEW> 800
NAM> EXIT
symbol "TEST1" defined as 00000500
symbol "TEST2" defined as 00000800
PATCH>
```

The DEFINE command assigns the symbols TEST1 and TEST2 to 500 and 800, respectively. The EXIT command is entered in response to the name prompt to terminate this level of prompting.

INSERT

Description

Use INSERT to insert VAX-11 MACRO instructions into specific locations within an image file. To use this command, you must specify the /INSTRUCTION qualifier or you must set the INSTRUCTION mode. To insert additional data into a patch area, use the DEPOSIT/PATCH_AREA command.

Before inserting the new instruction, the INSERT command confirms the contents of the location preceding the insertion (that is, that current instruction).

When the INSERT command is executed, it replaces the current instruction with a branch instruction and places the current instruction and the new instructions in the current patch area. The last new instruction is always followed by a branch instruction that redirects the flow of execution back to the inline code. The INSERT command automatically generates branch instructions.

After the insertion of new instructions, the patch area string descriptor is updated to reflect the modifications. See Section 5.3 for more information on patch area string descriptors.

Calculating the Location for the Branch Instruction: When the INSERT command is executed, it replaces the current instruction with a branch instruction and places the current instruction and the new instructions in the patch area. If the branch instruction to the patch area is longer than the current instruction, additional instructions following the current instruction are also moved to the patch area, and the branch instruction is deposited in the vacated memory locations. Unused memory locations are filled with NOP instructions.

On the other hand, if the current instruction is longer than the branch instruction, the unused memory locations are filled with NOP instructions.

Calculating Relative Displacements for Branch Instructions: PATCH calculates the relative displacements for the branch instructions it generates and recalculates the relative displacements for all branch-type instructions moved to the patch area. Instructions and data moved to patch area may, however, be referenced by instructions not affected by the move. Note that PATCH does not recalculate any relative displacements in the unaffected instructions.

Note also that if PATCH moves an instruction with a current address defined by a symbolic instruction label, you must check and correct any references made to that label. (See Section 4.1.5 for a complete description on symbolic instruction labels.)

Format

```
INSERT location = current-instruction
new-instruction
        .
        .
        .
```

Mode Qualifiers	Defaults
/OCTAL /DECIMAL /HEXADECIMAL	/HEXADECIMAL
/[NO]INSTRUCTION /[NO]SYMBOLS	/NOINSTRUCTION /SYMBOLS
/[NO]GLOBALS /[NO]SCOPE	/NOGLOBALS /SCOPE

Prompts

```
LOC> location
OLD> current-instruction
NEW> new-instruction
```

Parameters

location

Specifies the address after which one or more new instructions are to be added.

current-instruction

Specifies the instruction currently occupying the specified location.

new-instruction

Specifies one or more new instructions to be inserted into the image file following the current instruction.

Mode Qualifiers

/OCTAL
/DECIMAL
/HEXADECIMAL

Determine the radix in which virtual addresses are interpreted and displayed. The radix mode qualifiers do not affect symbolic instructions.

The initial default setting is /HEXADECIMAL.

/INSTRUCTION
/NOINSTRUCTION

Control whether the data to be inserted is interpreted as VAX-11 MACRO instructions. To use the INSERT command, you must specify the /INSTRUCTION mode qualifier or you must set the INSTRUCTION mode.

The initial default setting is /NOINSTRUCTION.

/SYMBOLS
/NOSYMBOLS

Control whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/GLOBALS
/NOGLOBALS

Control whether the symbolic entry (exactly as entered) is used as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/SCOPE
/NOSCOPE

Control whether scope's contribution to a pathname to find locations.

The initial default setting is /SCOPE.

Examples

1.
```
PATCH> SET MODE INSTRUCTION
PATCH> EXAMINE 600
00000600:     MOVL R1,R4
PATCH> EXAMINE
00000602:     BSBB LP_NAME
PATCH> INSERT 602 = 'BSBB LP_NAME'
NEW> 'CVTFW R3,#32'
NEW> EXIT
old:     602:     BSBB     LP_NAME
old:     604:     CMPB     R2,R8
new:     602:     BRW      PAA
new:     605:     NOP
new:     606:     NOP
new:     PAA:     BSBW     LP_NAME
new:     30374:     CVTFW    R3,#32
new:     30377:     CMPB     R2,R8
new:     30740:     BRW      607
```

This example shows the procedure used to insert an additional instruction into the existing code. After confirming the contents of location 602 and specifying the new instruction to be inserted, PATCH performs the following steps.

a. Determines how many instructions from the existing code must be moved to patch area to make room for a branch instruction.

b. Moves the instructions calculated in step 1 to the current patch area, and inserts in their places a branch instruction. PATCH calculates the relative displacement value for the branch instruction and generates a patch area symbol name to identify the destination of the branch.

c. Fills the unoccupied locations in the existing code with NOP instructions.

d. Recalculates the relative displacements for all branch-type instructions moved to patch area.

e. Inserts into the patch area a branch instruction that reroutes the program's flow of execution back to the inline code.

REPLACE

Description

Use REPLACE to replace the contents of one or more locations with new instructions or data in terms of the current mode settings. Before performing the replacement, the REPLACE command confirms the contents of the specified locations.

Replacing Instructions: When you replace instructions and the new instructions occupy more bytes in memory than the current instructions, the new instructions are moved to patch area and PATCH generates branch instructions to maintain the program's logical flow of execution. PATCH generates branch instructions for the REPLACE command the same way that it does for the INSERT command. (See the INSERT command description for a description of the mechanics of generating branch instructions.) All unused bytes are filled with NOP instructions.

If patch area is used to accommodate the new contents, the patch area string descriptor is updated to reflect the modifications. See Section 5.3 for more information on patch area string descriptors.

Replacing Data: When you replace ASCII or numeric data, the number of replacement entries cannot exceed the number of existing entries. This means, for example, that if you confirm the contents of six consecutive locations, you can replace the contents of only those six locations. If the number of replacement entries is less than the number of existing entries, the remaining locations are filled with zeros.

In addition, PATCH truncates replacement entries if they exceed the limit imposed on them by the current length mode. For ASCII characters, the right-most characters are discarded. For numeric data, the left-most digits are discarded.

Calculating Relative Displacements for Branch Instructions: PATCH calculates the relative displacements for the branch instructions it generates and recalculates the relative displacements for all branch-type instructions moved to the patch area. Instructions and data moved to the patch area may be referenced by instructions not affected by the move. Note that PATCH does not recalculate the relative displacement values in the unaffected instructions.

Note also that if PATCH moves an instruction with a current address defined as a symbolic label, you must check and correct any references made to that label. See Section 4.1.5 for a complete description of symbolic instruction labels. Confirms that one or more consecutive locations contain the contents supplied; then replaces the contents with new contents.

Format

```
REPLACE location = current-contents [,...]
new-contents
        .
        .
        .
```

Mode Qualifiers	Defaults
/BYTE /WORD /LONG	/LONG
/OCTAL /DECIMAL /HEXADECIMAL	/HEXADECIMAL
/[NO]ASCII /[NO]INSTRUCTION /[NO]SYMBOLS	/NOASCII /NOINSTRUCTION /SYMBOLS
/[NO]GLOBALS /[NO]SCOPE	/NOGLOBALS /SCOPE

Prompts

```
LOC> location
OLD> current-contents
NEW> new-contents
```

Parameters

location

Specifies either (1) a single location whose contents are to be replaced or (2) the starting address of a sequence of locations whose contents are to be replaced.

current-contents

Specifies one or more data entries or instructions to be replaced. The data or instructions you specify must be the actual contents.

Do not specify conflicting data types within a single REPLACE command.

new-contents

Specifies one or more data entries or instructions that are to replace the current contents.

Do not specify conflicting data types within a single REPLACE command.

Mode Qualifiers

/BYTE
/WORD
/LONG

Determine whether data is replaced in byte, word, or longword lengths. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the REPLACE command ignores the current length setting and replaces the entire instructions.

The initial default setting is /LONG.

/OCTAL
/DECIMAL
/HEXADECIMAL

Determine the radix in which numeric data and addresses are interpreted and displayed. The radix mode qualifiers do not affect symbolic addresses, symbolic instructions, or ASCII data.

The initial default setting is /HEXADECIMAL.

/ASCII
/NOASCII

Control whether the data to be replaced and the data to be deposited are interpreted and displayed as ASCII data.

When you specify /ASCII, enclose the data within matching quotation marks or apostrophes. The current radix setting has no affect on ASCII data being replaced or deposited; however, the REPLACE command truncates ASCII data if it exceeds the length imposed on it by the current length mode.

The initial default setting is /NOASCII.

/INSTRUCTION
/NOINSTRUCTION

Control whether the data to be replaced and the data to be deposited are interpreted and displayed as VAX-11 MACRO instructions.

When you specify /INSTRUCTION, enclose the instructions within matching quotation marks or apostrophes. Neither the current radix setting nor the current length setting affect the instruction being replaced..

The initial default setting is /NOINSTRUCTION.

/SYMBOLS
/NOSYMBOLS

Control whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/GLOBALS
/NOGLOBALS

Control whether the symbolic entry (exactly as entered) is used as the first or last pathname in a search. The initial default setting is /NOGLOBALS.

The initial default setting is /NOGLOBALS.

/SCOPE
/NOSCOPE

Control whether scope's contribution to a pathname is used to find locations.

The initial default setting is /SCOPE.

Examples

1.
```
PATCH> SET MODE INSTRUCTION, NOSYMBOLS
PATCH> REPLACE 600 = 'TSTL R4'
NEW> 'CMPB R2,R4'
NEW> EXIT
old:            00000600: TSTL R4
old:            00000602: BEQL 00000610
new:            00000600: BRW 0000784A
new:            00000604: NOP
new:
new:            0000784A: CMPB R2,R4
new:            0000784D: BNEQ 7852
new:            0000784F: BRW 00000610
                00007852: BRW 605
```

The instruction occupying location 600 is replaced with the instruction CMPB R2,R4. This instruction occupies more bytes than the instruction TSTL R4. Thus, to make room for CMPB R2,R4, the instructions CMPB R2,R4 and the instruction that follows TSTL R4 are moved to the current patch area. A branch instruction is deposited in place of TSTL R4 to direct program execution to the patch area. The last instruction deposited in the patch area is a branch instruction back to the inline code.

2.
```
PATCH> SET SCOPE MOD1
PATCH> REPLACE/SCOPE RO_CODE+20 = 1000
NEW> 100
NEW> EXIT
old:            MOD1\E: 00001000
new:            MOD1\E: 00000100
```

The SET SCOPE command establishes the current symbolic scope as MOD1. A subsequent REPLACE command requests that the contents of location RO_CODE+20 in the module named MOD1 be replaced with new data. The display indicates that the replacement was successful. Note that the location is reported by the pathname MOD1\E because the symbol E is the closest symbol to location RO_CODE+20.

SET ECO

SET ECO

Description

Use SET ECO to define the ECO level for the ensuing patch.

Whenever you apply a patch to an image file, the first command you type should be a SET ECO command. This command provides you with a way to identify your patches easily. More important, however, it lets you process selected patches using a command procedure. When PATCH processes a command procedure, it searches the file for ECO levels and processes only those patches represented by the specified ECO levels. Patches not represented by ECO levels are not processed.

You can issue as many SET ECO commands as necessary during a PATCH session; however, once you specify an ECO level, that level can never be used again for that particular image file. Furthermore, whenever you begin a patch with a SET ECO command, you must also terminate the patch with the UPDATE command. If you fail to issue the UPDATE command (1) the ECO level specified with the SET ECO command is not set (hence, the patch is not applied to the image file), and (2) you cannot issue another SET ECO command.

Format

SET ECO eco-level

Prompts

ECO> eco-level

Parameters

eco-level

Specifies a decimal integer between 1 and 128, inclusive. ECO levels outside this range of integers are illegal.

Examples

1.
```
PATCH> SET ECO 15
    .
    .
    .
PATCH> UPDATE
%PATCH-I-WRTFIL, updating image file DBA3:[HOWARD]NUFILE.EXE;2
PATCH> CHECK NOT ECO 15
%PATCH-E-ECOSET, eco level 15 already set in DBA3:[HOWARD]NUFILE.EXE;2
```

The SET ECO command defines the ECO level for the subsequent patch as 15. The commands are then entered. When the UPDATE command is issued, the ECO level is set and the image file is updated to include the affects of the new commands. The CHECK NOT ECO indicates that ECO level 15 is set in the image file NUFILE.EXE.

SET MODE

Description

Use SET MODE to control the syntax of the commands you enter and the values PATCH displays.

Format

SET MODE mode [,...]

Prompts

NEW> mode

Parameters

mode

Specifies one or more modes from the context, radix, length, and symbol search mode categories to be established as the current modes. These modes determine how PATCH interprets entries and displays output.

The commands that are affected by the entry and display modes are: DELETE, DEPOSIT, EVALUATE, EXAMINE, INSERT, REPLACE, and VERIFY.

Table 8-1 summarizes the modes that can be set. See Chapter 6 for a complete description of the rules for specifying entry and display modes.

Table 8-1
Mode Values for the SET MODE Command

Category	Mode	Description
Context	ASCII NOASCII	Control whether data is accepted and displayed as ASCII characters
	INSTRUCTION NOINSTRUCTION	Control whether data is accepted and displayed as VAX-11 MACRO instructions.
	SYMBOLS NOSYMBOLS	Control whether addresses are reported by pathnames or symbols rather than numeric addresses
Radix	OCTAL DECIMAL HEXADECIMAL	Determine the base in which numeric addresses and data are interpreted and displayed

(continued on next page)

Table 8-1 (Cont.)
Mode Values for the SET MODE Command

Category	Mode	Description
Length	BYTE WORD LONG	Determine the length in which numeric data is accepted and displayed
Symbol Search	GLOBALS NOGLOBALS	Control whether the symbolic entry (exactly as entered) is used as the first or last pathname in a search.
	SCOPE NOSCOPE	Control whether scope's contribution to a pathname is used to find locations.

Also note that you can alter a mode setting by specifying a mode qualifier with a command.

Examples

1.
```
PATCH> SET MODE INSTRUCTION,GLOBALS,NOSYMBOLS
```

This command requests that the modes INSTRUCTION, GLOBALS, and NOSYMBOLS be established as the current modes.

2.
```
PATCH> SET MODE INSTRUCTION
PATCH> DEPOSIT 78B = 'BEQL NEWPATCH'
```

This SET MODE command establishes INSTRUCTION mode as a current mode setting. A subsequent DEPOSIT command can then deposit an instruction without specifying the /INSTRUCTION mode qualifier.

3.
```
PATCH> SET MODE
NEW> OCTAL
NEW> WORD
NEW> EXIT
```

In this example, the SET MODE continuously prompts for new modes until EXIT is typed.

SET MODULE

Description

Use SET MODULE to enter local symbol information from the specified modules into PATCH's symbol table, provided you followed the rules for passing local symbol information to PATCH (See Chapter 4).

If the symbol table is too small to accommodate the local symbol information, PATCH displays an error message.

Note that if you are patching a shareable image, no local or global symbol information is passed to PATCH; only universal symbols can be accessed. (See Chapter 4 for a complete description on using universal symbols at PATCH time and read the VAX-11 Linker Reference Manual for information on declaring symbols as universal at link time.)

Format

SET MODULE module-name [,...]

Other Qualifiers	Defaults
/ALL	None.

Prompts

NAM> module-name

Parameters

module-name

Specifies the name of one or more modules whose local symbols are to be entered in the symbol table.

Do not specify a module name if you include the /ALL qualifier.

Other Qualifiers

/ALL

Requests that local symbol information from all the modules in the image file be entered in the symbol table.

Examples

1.
```
PATCH> SET MODULE   GREAT_APES
PATCH> EXAMINE/ASCII ORANGUTANS:ORANGUTANS+8
GREAT_APES\ORANGUTANS:  'AMAZ'
GREAT_APES\ORANGUTANS+4:  'ING '
```

THE SET MODULE command enters all local symbol information in the module GREAT_APES into the symbol table. A subsequent EXAMINE command can use the local symbol ORANGUTANS to reference particular locations.

2.
```
PATCH> SET MODULE/ALL
```

This command requests that all symbol information from all modules be added to the symbol table.

SET PATCH_AREA

Description

Use SET PATCH_AREA to override the use of PATCH's default patch area in favor of the patch area you defined at assembly or compile time. (Section 5.2 describes the procedure for creating user-defined patch area.)

Whenever you use a user-defined patch area, its patch area descriptor is updated to reflect any modifications.

Format

SET PATCH_AREA address-of-string-descriptor

Prompts

NEW> address-of-string-descriptor

Parameters

address-of-string-descriptor

Defines the address of the patch area descriptor. The patch area descriptor can be represented as a symbolic or a numeric address. See Section 5.3 for a complete description of a patch area descriptor.

Examples

1.
```
PATCH> SET PATCH_AREA LRDPATCHES
PATCH> SHOW PATCH_AREA
current patch area size:    0000004F
current patch area address: 000005F2
```

The SET PATCH_AREA command establishes the user-defined patch area LRDPATCHES as the current patch area. The SHOW PATCH_AREA command reports the current status of LRDPATCHES. With this information, you can deposit instructions or data into LRDPATCHES.

SET SCOPE

Description

Use SET SCOPE to establish the specified module name (and routine name, if specified) as the explicit scope to be used under the rules for translating pathnames and symbols into values. These rules are discussed in Chapter 4.

PATCH also inserts local symbol information associated with the specified module into the symbol table when you type the SET SCOPE. If the symbol table is too small to accommodate this information, PATCH issues an error message.

Use the SHOW MODULE command to determine the modules to which the scope can be set.

If the local symbols in a specific module have not been entered in the symbol table by the SET MODULE command, the SET SCOPE command forces those symbols into the table.

Format

SET SCOPE module-name [\routine-name [\...]]

Prompts

NEW> module-name [\routine-name]

Parameters

module-name

Specifies the name of the module to which the scope is to be set.

routine-name

Specifies the name of a routine contained in the module to which the scope is to be set.

Examples

1. PATCH> SET SCOPE CARP_LPT

This set command establishes the scope as CARP_LPT.

2. PATCH> SET SCOPE
   NEW> LIP_SIGN
   %PATCH-W-NOSUCHMODU, no such module name "LIP_SIGN"

In this example, there is no such module named LIP_SIGN. Therefore, the SET SCOPE command fails and the previous scope setting is retained.

SHOW MODE

Description

Use SHOW MODE to report the modes that are currently set.

This command is used primarily with the SET MODE and/or CANCEL MODE commands. It indicates the condition of the current modes, enabling you to change one or more of them if necessary.

When you issue the SHOW MODE command, the mode values are always displayed in lowercase letters.

Format

SHOW MODE

Prompts

None.

Parameters

None.

Examples

1.
```
PATCH> SHOW MODE
modes:  symbols, noinstruction, noascii, scope, noglobals, hexadecimal long
```

The SHOW MODE command requests that the current modes be displayed.

SHOW MODULE

Description

Use SHOW MODULE to request PATCH to display all the modules in the image file and indicate whether the symbols contained in the modules are available for use. It also indicates the amount of symbol table space required by each module and routine and the total amount of unused space remaining in the symbol table.

The SHOW MODULE command reports an informational error message if no local symbol information was passed to PATCH.

Format

SHOW MODULE

Prompts

None.

Parameters

None.

Examples

1.
```
PATCH> SHOW MODULE
module name     symbols     size

AVERAGE            no         52.
OTS$LINKAGE        no         128.

total modules:     2.
remaining size:    64052.
PATCH> SET MODULE/ALL
PATCH> SHOW MODULE
module name     symbols   size

AVERAGE            yes        52.
OTS$LINKAGE        yes        128.

total modules:     2.
remaining size:    63880.
```

The first SHOW MODULE command indicates that there are two modules in the current image file, though neither one has its local symbols entered in the symbol table. The SET MODULE command requests that all local symbol information be entered in the symbol table. The second SHOW MODULE command confirms the presence of the local symbols in the symbol table.

SHOW PATCH_AREA

Description

Use SHOW PATCH_AREA to report the size and starting address of the current patch area in hexadecimal representation, regardless of the current radix setting. Generally, you use this command with either the ALIGN command or the DEPOSIT/PATCH_AREA command. In both cases, you want to know the status of the patch area before realigning it or depositing data in it.

Format

SHOW PATCH_AREA

Prompts

None.

Parameters

None.

Example

1.
```
PATCH> SHOW PATCH_AREA
current patch area size:   00000030
current patch area address:  00000308
DEPOSIT/PATCH_AREA/ASCII   308= 'AT'
old:  00000308:
new:  00000308:  'AT'
```

The SHOW PATCH_AREA display reports that the current patch area size is 30 bytes and the starting address is memory location 308. A subsequent DEPOSIT command can deposit data into the patch area.

SHOW SCOPE

Description

Use SHOW SCOPE to display the current scope setting.

Format

SHOW SCOPE

Prompts

None.

Parameters

None.

Examples

1.
```
PATCH> SHOW SCOPE
scope:  TEMP1
PATCH> EXAMINE DEGREES+8
```

The SHOW SCOPE command indicates that the current scope setting is TEMP1. A subsequent EXAMINE command can reference a location using local symbol information contained in TEMP1.

UPDATE

Description

Use UPDATE to apply a patch to the image file.

The UPDATE command is a patch terminator. This command applies the previous patch to the image file and thus creates an output image file. If you fail to issue the UPDATE command, no output image file is created.

During a single execution of PATCH, you can specify the UPDATE command more than once. The first UPDATE command specified creates a new output image file. Subsequent UPDATE commands (specified during that PATCH session) overwrite the output image file. That is, a new version of the output image file is not created.

If a SET ECO command is issued, a subsequent UPDATE command automatically sets the ECO level specified. (See the description of the SET ECO command.)

Format

UPDATE

Prompts

None.

Parameters

None.

Examples

1.

```
        .
        .
        .
PATCH> UPDATE
%PATCH-I-WRTFIL, updating image file DB1:[MASON]HOROSCOPE.EXE;4
```

The UPDATE command applies the previous patch to the image file HOROSCOPE.EXE. The UPDATE display indicates that the image file has been successfully updated.

VERIFY

Description

Use VERIFY to confirm that a location or several consecutive locations contain the specified contents (instructions or data).

Generally, you should use the VERIFY command with the DEPOSIT command. The DEPOSIT command does not confirm the entries before overwriting them. Therefore, to make certain you know the contents of the locations that will be modified, issue the VERIFY command first.

The VERIFY command is also useful when you are patching an image file by means of a command procedure. Using the VERIFY command, you can check particular locations before attempting to modify them. If the VERIFY command fails, that patch is not applied, and PATCH skips to the next SET ECO command. If no other SET ECO command exists, the command procedure is terminated.

Format

```
VERIFY location = current-contents [,...]
```

Mode Qualifiers	Defaults
/BYTE /WORD /LONG	/LONG
/OCTAL /DECIMAL /HEXADECIMAL	/HEXADECIMAL
/[NO]ASCII /[NO]INSTRUCTION /[NO]SYMBOLS	/NOASCII /NOINSTRUCTION /SYMBOLS
/[NO]GLOBALS /[NO]SCOPE	/NOGLOBALS /SCOPE

Prompts

```
LOC> location
OLD> current-contents
```

Parameters

location

Specifies either (1) a single location whose contents are to be checked or (2) the starting address of a sequence of locations whose contents are to be checked.

current-contents

Specifies one or more data entries or instructions to be verified. The data or instructions you verify must be the actual contents.

Do not specify conflicting data types within a single VERIFY command.

Mode Qualifiers

/BYTE
/WORD
/LONG

Determine whether data is verified in byte, word, or longword lengths. The length mode qualifiers affect only ASCII and numeric data. When you specify an instruction, the VERIFY command ignores the current length mode and verifies the entire instruction.

The initial default setting is /LONG.

/OCTAL
/DECIMAL
/HEXADECIMAL

Determine the radix in which numeric data and addresses are interpreted and displayed. The radix mode qualifiers do not affect symbolic addresses, symbolic instructions, and ASCII data.

The initial default setting is /HEXADECIMAL.

/ASCII
/NOASCII

Control whether the data to be verified is interpreted and displayed as ASCII data.

When you specify /ASCII, enclose the data within matching quotation marks or apostrophes. The current radix setting has no affect on the ASCII data being verified; however, the VERIFY command truncates the data if it exceeds the limit imposed on it by the current length mode.

The initial default setting is /NOASCII.

/INSTRUCTION
/NOINSTRUCTION

Control whether the data to be verified is interpreted and displayed as VAX-11 MACRO instructions.

When you specify /INSTRUCTION, enclose the instructions within matching quotation marks or apostrophes. Neither the current radix setting nor the current length setting affect the instruction being verified.

The initial default setting is /NOINSTRUCTION.

/SYMBOLS
/NOSYMBOLS

Control whether locations are displayed as pathnames or symbols, rather than as numeric addresses.

The initial default setting is /SYMBOLS.

/GLOBALS
/NOGLOBALS

Control whether the symbolic entry (exactly as entered) is used as the first or last pathname in a search.

The initial default setting is /NOGLOBALS.

/SCOPE
/NOSCOPE

Control whether scope's contribution to a pathname is used to find the specified location.

The initial default setting is /SCOPE.

Examples

1.
```
PATCH> VERIFY/ASCII/NOSYMBOLS 6FF = 'RAIN'
old:   000006FF:   'RAIN'
```

The VERIFY command confirms that the ASCII text RAIN resides in location 6FF. The /NOSYMBOLS mode qualifier requests that location 6FF be displayed as a virtual address in terms of the current radix.

APPENDIX A

COMMAND SUMMARY

This appendix lists in alphabetical order each PATCH command in its required format, the qualifiers you can specify with the command, and a brief description of the command's function. The bold face letters indicate the characters you must type for PATCH to recognize the command name or qualifier.

Format	Function
ALIGN /alignment-qual symbol **/BY**TE, **/WO**RD, **/LO**NG, **/QU**AD, **/PA**GE	Aligns the current patch area on the given boundary and assigns a symbol to that address.
CANCEL **MO**DE	Cancels the current modes and reinstates the initial default modes.
CANCEL **MODU**LE module-name [,...] **/ALL**	Removes local symbols defined in one or more given modules from PATCH's symbol table.
CANCEL **PA**TCH_AREA	Cancels the use of the current patch area and reverts to the use of the default patch area.
CANCEL **SC**OPE	Cancels the current symbolic scope and reverts to the null scope.
CHECK **EC**O eco-level [:...] [,...]	Verifies the application of the patches associated with the given ECO levels.
CHECK **NOT EC**O eco-level [:...] [,...]	Verifies that the given ECO levels are available for use.
CREATE [file-spec]	Creates a command procedure. All subsequent, successful commands are written into this file.
DEFINE symbol = value [,symbol = value,...]	Assigns a symbol to a value. Multiple assignments can be made.
DELETE /mode-qual loc = current-contents [,...] **/BY**TE, **/WO**RD, **/LO**NG, **/OC**TAL, **/DEC**IMAL, /HEXADECIMAL, **/[NO]AS**CII, **/[NO]**INSTRUCTION, **/[NO]GL**OBALS, **/[NO]SC**OPE, **/[NO]SY**MBOLS	Verifies the current contents of one or more consecutive locations, then replaces the contents with NOPs or zeros.
DEPOSIT /mode-qual /add-qual loc = new-contents [,...] **/BY**TE, **/WO**RD, **/LO**NG, **/OC**TAL, **/DEC**IMAL, /HEXADECIMAL, **/[NO]AS**CII, **/[NO]**INSTRUCTION, **/[NO]GL**OBALS, **/[NO]SC**OPE, **/[NO]SY**MBOLS, **/PA**TCH_AREA	1. Deposits new contents into one or more consecutive locations. 2. Deposits new contents into one or more consecutive locations in the current patch area. Use the /PATCH_AREA qualifier.
EVALUATE /mode-qual expression [,...] **/BY**TE, **/WO**RD, **/LO**NG, **/OC**TAL, **/DEC**IMAL, /HEXADECIMAL, **/[NO]AS**CII, **/[NO]**INSTRUCTION, **/[NO]GL**OBALS, **/[NO]SC**OPE, **/[NO]SY**MBOLS	Displays the current definition of one or more expressions.

COMMAND SUMMARY

Format	Function
EXAMINE /mode-qual loc [:loc] [,...] /**BYTE**, /**WORD**, /**LONG**, /**OCTAL**, /**DEC**IMAL, /**HEXADECIMAL**, /**[NO]ASCII**, /**[NO]**INSTRUCTION, /**[NO]GLOBALS**, /**[NO]SCOPE**, /**[NO]SYMBOLS**	Displays the contents of one or more locations.
EXIT	1. Terminates a command sequence prompt. 2. Terminates a PATCH session.
INSERT /mode-qual loc = current-instruction new-instruct . . . /**OCTAL**, /**DEC**IMAL, /**HEXADECIMAL**, /INSTRUCTION, /**[NO]GLOBALS**, /**[NO]SCOPE**, /**[NO]SYMBOLS**	Verifies the current contents of a location, then inserts one or more symbolic instructions after that location.
REPLACE /mode-qual loc = current-contents [,...] new-contents . . . /**BYTE**, /**WORD**, /**LONG**, /**OCTAL**, /**DEC**IMAL, /**HEXADECIMAL**, /**[NO]ASCII**, /**[NO]** INSTRUCTION, /**[NO]GLOBALS**, /**[NO]SCOPE**, /**[NO]SYMBOLS**	Verifies the contents of one or more consecutive locations, then replaces the contents with new contents.
SET ECO eco-level	Defines an ECO level for the ensuing patch. The ECO level is not set until an UPDATE command is issued.
SET MODE mode-parameter [,...]	Sets one or more new entry and display modes.
SET MODULE module-name [,...] /ALL	Enters local symbol information defined in one or more given modules into PATCH's symbol table.
SET PATCH_AREA address-of-string-descriptor	Establishes a user-defined patch area as the current patch area.
SET SCOPE module-name \routine-name [\...]	Defines the current symbolic scope.
SHOW MODE	Displays the current mode settings.
SHOW MODULE	Displays all modules in the image that is being patched and displays whether the local symbols contained in those modules are available for use.
SHOW PATCH_AREA	Displays the string descriptor of the current patch area.
SHOW SCOPE	Displays the current symbolic scope.
UPDATE	Sets the ECO level and applies the patch to the image file.
VERIFY /mode-qual loc = current-contents [,...] /**BYTE**, /**WORD**, /**LONG**, /**OCTAL**, /**DEC**IMAL, /**HEXADECIMAL**, /**[NO]ASCII**, /**[NO]**INSTRUCTION, /**[NO]GLOBALS**, /**[NO]SCOPE**, /**[NO]SYMBOLS**	Verifies the contents of one or more consecutive locations.

APPENDIX B

ERROR MESSAGES

When you issue an incorrect PATCH command, PATCH responds by displaying a descriptive error message. All error messages are reported to SYS$ERROR and SYS$OUTPUT. Error messages are also recorded in the journal file corresponding to the image file being patched. This lets you examine all the commands you used to patch your image file.

Error messages are divided into four categories: informational messages, warning messages, severe error messages, and fatal error messages. These categories are distinguished by their level of severity. When processing command procedures, PATCH terminates execution and aborts the procedure if it encounters an error other than an informational message. During an interactive session, PATCH is aborted only upon encountering a fatal error.

The format for reporting error messages is as follows:

%FACILITY-L-TTTTT, message

%FACILITY

The facility issuing the error message. In most cases the facility will be PATCH.

L

The severity level of the error. The severity levels are:

I -- Informational message
W -- Warning message
E -- Severe error message
F -- Fatal error message

TTTTT:

A mnemonic description of the error.

message

The error message displayed in response to the erroneous PATCH command.

The remainder of this chapter lists and describes the error messages issued by PATCH. The error messages are listed alphabetically, regardless of the severity level. This is because the severity level for a particular error message may differ depending upon the context in which it was issued.

BADECO, illegal eco level of nn in file-spec

Indicates that you specified an ECO level outside the range of legal ECO levels. Legal ECO levels fall between 1 and 128, inclusive.

BADIDENT, unable to patch image with ident of nn

Indicates that PATCH does not understand this version of the image file. Relink your image file.

BADOPCODE, opcode xx is unrecognized

Indicates that you specified an instruction that PATCH does not recognize. Note that PATCH does not accept abbreviations for instructions. For example, ADDL cannot be substituted for ADDL2.

BADSCP, scope setting not module or routine

Indicates that the symbol you specified as the new scope value is not a module name or routine name.

BRTOOFAR, destination addr is too far for branch operand

Indicates that the relative displacement for the branch-type instruction is not large enough to reach the specified destination address.

CLIERR, cli error parsing command line

Indicates that you incorrectly entered the command line that invokes PATCH; reenter the command line correctly.

CLSERR, close error code of xx on file file-spec

Indicates that the $CLOSE request failed due to the error code represented as xx.

CNTERR, connect error code of xx on file file-spec

Indicates that a VAX-11 RMS $CONNECT request failed due to the error code represented as xx.

CRMPSC, error code of xx mapping section

Indicates that the system service $CRMPSC request failed due to the error represented as xx.

DELTVA, error code of xx deleting xxx to xxx

Indicates that the system service $DELTVA request failed due to the error code represented as xx.

DIFVAL, memory contains different value than specified

Indicates that the value you specified is not contained within the given memory location.

DIVZERO, attempt to divide by zero

Indicates that you attempted to divide by zero.

DUPCOMFIL, command file file-spec already being written

Indicates that you have already requested that a command procedure be written for this PATCH session.

ECONOTSET, eco level nn not set in file-spec

Indicates that you specified an ECO level that has not yet been set in that particular image file. This typically occurs when you issue the CHECK ECO command before issuing the UPDATE command.

ECOSET, eco level nn already set in file-spec

Indicates that you specified an ECO level that has already been set in that particular image file.

ENDPRS, error code of xx on command line parse end

Indicates that a command interpreter callback request failed due to the error code represented as xx.

EXARANGE, invalid range of addresses

Indicates that you specified a range of addresses using an incorrect format. The correct format is:

lower-boundary-address:upper-boundary-address

EXPREG, error code of xx expanding program region

Indicates that the system service $EXPREG request failed due to the error code represented as xx.

EXPSTKOVR, expression too nested, stack overflow

Indicates that you attempted to evaluate an expression that contains more than 16 levels of nested arguments.

EXTBIT, illegal extract bit number - greater than 31

Indicates that you specified an illegal bit number; legal bit numbers range from 0 through 31, inclusive.

FRERANGE, storage package range error

Indicates an internal PATCH error; please submit a Software Performance Report.

FRESIZE, storage package size error

Indicates an internal PATCH error; please submit a Software Performance Report.

GBLONLY, cannot obtain global copy of image section image-section

Indicates that the image being patched references a global copy of an image section. There is no local copy of this section within the image file. To patch such a section, you must patch the shareable image file in which it is contained.

GBLWARN, patching local copy of global image section image-section

Indicates that the image being patched references a global image section. The image contained a local copy of the global image section and only the local copy is being patched.

GETERR, read error code of xx on file file-spec

Indicates that a VAX-11 RMS $GET read request failed due to the error code represented as xx.

GETTIM, error code of xx on $GETTIM

Indicates that the system service $GETTIM request failed due to the error code represented as xx.

HDRBLK, too many isd's for image header blocks

Indicates that you requested too many image section descriptors; PATCH is unable to create patch area. Relink your image file.

INITPRS, error code of xx on command line parse init

Indicates a CLI callback failure due to the error code represented as xx; please submit a Software Performance Report.

INSOPRND, missing instruction operand

Indicates that you omitted one or more operands from the instruction.

INSUFPAT, nn bytes requested, patch area at addr contains nn bytes

Indicates that the user-defined patch area is not large enough to accommodate the specified code. To remedy the situation, use an alternate patch area, for example, the default patch area. Generally, the default patch area is expandable.

INVCMD, invalid command

Indicates that you issued a command and either omitted or incorrectly entered required punctuation.

INVDSTREC, invalid DST record

Indicates an error in the debugger's symbol table of that particular image file; please submit a Software Performance Report.

INVIMGHDR, inconsistent debug and symbol data in image header

Indicates that the image header inconsistently describes the symbol information causing the absence of symbol information at PATCH time. Please submit a Software Performance Report.

INVNUMBER, invalid numeric string

Indicates that you specified a numeric data string that is illegal in the current context.

INVOPR, unrecognized operator in expression

Indicates that the expression contained a character that PATCH did not recognize, in place of a valid operator.

INVOPRND, unable to resolve operand xx

Indicates that you specified an operand that PATCH does not recognize. Operands such as 4(r0) must be preceded by B^, for example, B^4(R0).

INVQUO, quoted argument not allowed in current mode

Indicates that you attempted to enter an argument enclosed in quotation marks or apostrophes and you were not in INSTRUCTION or ASCII mode.

INVTOKEN, unrecognized token symbol-name

Indicates that the symbol you specified is not recognized as a valid PATCH command name or qualifier.

JNLPUT, error writing to journal file

Indicates that an output error occurred while PATCH was writing to the journal file.

LONGDST, unable to include all module names in symbol table

Indicates that when you initialized PATCH, the image file could not accommodate all the modules it found. To determine which modules were placed in the symbol table, issue the SHOW MODULE command. To make room for the new modules, issue the CANCEL MODULE command specifying the module names whose symbols are no longer required. Then reissue the SET MODULE command.

MEMBUG, dynmem error size=nn addr=nn error=nn

Indicates an internal PATCH error; please submit a Software Performance Report.

MISSQUO, unmatched quotation marks in argument

Indicates that you did not use matching quotation marks or apostrophes to enclose an argument. PATCH inserts the necessary quote for you and continues command execution.

MODIFYERR, modify error code of xx on file file-spec

Indicates that a VAX-11 RMS $MODIFY request failed due to the error code represented as xx.

MODNOTADD, insufficient symbol table space to add module module-name

Indicates that the SET MODULE command failed. This typically occurs when there is not enough space in the symbol table to accommodate the additional module or modules. To make room for the new modules, issue the SHOW MODULE command to determine which modules are in the symbol, then CANCEL MODULE command specifying the module names whose symbols are no longer required. Then reissue the SET MODULE command.

MULTECO, multiple eco levels (nn and nn) set for one patch

Indicates that you attempted to define a second ECO level before setting the first ECO level. To set the first ECO level, issue the UPDATE command.

MULTOPR, multiple successive operators in expression

Indicates that you entered an expression containing more than one operator in succession.

NOACCESS, cannot access image section

Indicates that you specified an address contained in an image section that is not within this image file.

NOANGLE, unmatched angle brackets in expression

Indicates that you entered an expression that has an unequal number of right angle brackets to left angle brackets.

NOBRANCH, instruction requires branch-type operand

Indicates that you attempted to enter a branch-type instruction with an invalid operand for a destination field. For example, PATCH does not accept: DEPOSIT/INSTRUCTION GO= 'BNEQ R0'.

NOCONTIG, non-contiguous image file being written

Indicates that the output file is a noncontiguous image file. To ensure a contiguous output file, after you exit from PATCH use:

```
$ COPY/CONTIGUOUS input-file-spec output-file-spec
```

NODECODE, cannot decode instruction

Indicates that PATCH encounterd an undefined opcode or illegal addressing mode while attempting to decode the instruction.

NOENCODE, cannot encode instruction instr

Indicates that the instruction contained an illegal opcode or operand. The branch destination may not be reachable from the location for which it was destined.

NOFREE, no free storage available

Indicates an internal PATCH error; please submit a Software Performance Report.

NOGBL, some or all global symbols not accessible

Indicates that when you initialized PATCH, no global symbol information was available to PATCH. To ensure that global symbol information is passed to PATCH, reassemble or recompile your program including the debug or traceback facility.

Or, the symbol table was not large enough to incorporate all the global symbol information.

NOLCL, image does not contain local symbols

Indicates that when you initialized PATCH, no local symbol information was available to PATCH. To ensure that local symbol information is passed to PATCH, reassemble or recompile your program including the debug or traceback facility.

NOLITERAL, no LITERAL translation exists for xx

Indicates that the value represented as xx does not have a symbolic equivalent associated with it.

NOOPRND, missing operand in expression

Indicates that you omitted one or more operands from the expression.

NOPATAREA, insufficient patch area at addr, size = nn

Indicates that the current user-defined patch area was not large enough to store the given code. Use the default patch area or an alternate user-defined patch area of a larger size.

NORELOC, relocation of CASE instructions not supported

Indicates that PATCH does not automatically move CASE instructions to the patch area. You move CASE instructions to the patch area by use of the DELETE/INSTRUCTION and DEPOSIT/INSTRUCTION/PATCH_AREA commands and the MACRO assembler directive .WORD.

NORSTBLD, cannot build symbol table

Indicates that when you initialized PATCH, it could not allocate space to build a symbol table.

NOSUCHMODU, no such module name module-name

Indicates that you specified the name of a module not contained in the image file.

NOSUCHSYM, no such symbol

Indicates that you specified a symbol not contained in PATCH's symbol table.

NOSYMBOL, no such symbol symbol-name

Indicates that you specified a symbol not contained in PATCH's symbol table.

NOTALLSYM, unable to initialize symbols for default modules

Indicates that the symbol table did not have enough room to include symbols for all the modules normally inserted into the table upon execution of PATCH.

NOTDONE, xx not yet a supported feature

Indicates that the feature you attempted to use is recognized but not implemented by PATCH. The name of the feature is indicated in the error message.

NOTPATAREA, patch area is addr1 not addr2

Indicates that you typed the incorrect starting address of the patch area. To determine the correct starting address, issue the SHOW PATCH_AREA command or the ALIGN/BYTE command. When you issue ALIGN/BYTE, PATCH defines a name as the current starting address.

NOTQUO, unquoted argument not allowed in ASCII or INSTRUCTION modes

Indicates that, while in ASCII or INSTRUCTION mode, you entered an argument not enclosed within matching apostrophes or quotation marks.

NOUPDATE, no patch for update qualifier eco level nn

Indicates that when you submitted a command procedure for interactive or batch execution (by typing PATCH/UPDATE = (eco-level, ...)), and you specified an ECO level that was not included in the command procedure.

NSADDR, address addr is not within image

Indicates that you specified an address outside the address range of the image file.

NUMOPRNDS, xx instructions must have nn operands

Indicates that you omitted an operand from the MACRO-11 instruction. Reissue the PATCH command with the correct number of operands.

NUMTRUNC, number truncated

Indicates that PATCH truncated a numeric data entry because it exceeded the current length mode.

NUMVAL, error code of xx requesting update value

Indicates that a command interpreter callback request failed due to the error code represented as xx.

OLDINSCHK, memory location addr not equal to xx

Indicates that the verification of the contents of the given address failed. Reissue the PATCH command specifying the correct value.

OLDVALCHK, memory location addr not equal to xx

Indicates that the verification of the contents of the given address failed. Reissue the PATCH command specifying the correct value.

OPNERR, open error code of xx on file file-spec

Indicates that a VAX-11 RMS $OPEN request failed due to the error code represented as xx.

OPRNDLNG, operand xx too long for internal buffers

Indicates an internal PATCH error; please submit a Software Performance Report.

OPSYNTAX, instruction operand syntax error

Indicates that there is a syntactical error in an operand within an instruction.

OUTCMDLNG, output command line too long for buffer

Indicates an internal PATCH error; please submit a Software Performance Report.

PARSEERR, internal parsing error

Indicates an internal PATCH error; please submit a Software Performance Report.

PATERR, internal patch coding error

Indicates an internal PATCH error; please submit a Software Performance Report.

PATHLONG, too many qualifiers on name

Indicates that you entered a pathname consisting of more than 15 elements.

PRIMIN, error code of xx parsing input file

Indicates that a command interpreter callback request failed due to the error code represented as xx.

PRIMOUT, error code of xx parsing journal file

Indicates that a command interpreter callback request failed due to the error code represented as xx.

PUTERR, put error code of xx on file file-spec

Indicates that an error writing the specified file occurred due to the error code represented as xx.

REDEFSYM, redefining symbol symbol-name from addr1 to addr2

Indicates that a user-defined symbol has been changed to a different value. This typically occurs when you issue the DEFINE command and specify a symbol currently used in the image file or when PATCH moves a VAX-11 MACRO instruction that contains a label to a patch area.

REPLACEERR, replacement value too large for location

Indicates that when you replaced data with the REPLACE command, the new data was comprised of more bytes than the old data.

RESOPCODE, opcode xx is reserved

Indicates that you specified an opcode that is reserved for DIGITAL use only.

SECOUT, error code of xx parsing journal file

Indicates that a command interpreter callback request failed due to the error code represented as xx.

STGTRUNC, string truncated

Indicates that PATCH truncated an ASCII string entry because it either exceeded the current length mode or was inappropriate for the context in which it was specified.

SYNTAX, command syntax error at or near xx

Indicates that you incorrectly entered the part of the command line represented as xx.

TEROUT, error code of xx parsing command file

Indicates that a command interpreter callback request failed due to the error code represented as xx.

UPDATE, patch with eco level nn ignored due to update qualifier

Indicates that PATCH ignored the patch associated with this ECO level because you used the /UPDATE qualifier and did not include this particular ECO level in the qualifier's value list.

WRTFIL, updating image file file-spec

Indicates that the specified image file has been updated to include the previous patch.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

— — — — Do Not Tear - Fold Here and Tape —

digital

No Postag
Necessar
if Mailed in
United Sta

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS TW/A14
DIGITAL EQUIPMENT CORPORATION
1925 ANDOVER STREET
TEWKSBURY, MASSACHUSETTS 01876

— — — Do Not Tear - Fold Here —